AFIT/GA/ENY/92D-10

NONLINEAR LARGE DISPLACEMENT AND MODERATE ROTATIONAL CHARACTERISTICS OF COMPOSITE BEAMS INCORPORATING TRANSVERSE SHEAR STRAIN

THESIS

Stephen G. Creaghan
Captain, USAF

AFIT/GA/ENY/92D-10

AFIT/GA/ENY/92D-10

NONLINEAR LARGE DISPLACEMENT AND MODERATE ROTATIONAL CHARACTERISTICS OF COMPOSITE BEAMS INCORPORATING TRANSVERSE SHEAR STRAIN

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Astronautical Engineering

Stephen G. Creaghan, B.S.C.E
Captain, USAF

Dec, 1992

Approved for public release; distribution unlimited

For guiding me through this thesis and keeping me on course, I thank my thesis advisor, Dr Anthony Palazotto. When aspects of shell theory were beyond me, Dr Palazotto was forever patient and never hesitated to explain a fine point one more time. To me, he was a great teacher. Thanks also to Captain Scott Schimmels who assembled and ran the MACSYMA decks necessary to genarate the stiffness matrices as FORTRAN code. Finally, thanks to my lovely wife, Marisa. She sacrificed many weekends during our engagement and was a beacon of love and encouragement throughout this effort.

Table of Contents

List of Figures

List of Tables

List of Symbols

AFIT/GA/ENY/92D-10

Abstract

This research was directed toward the investigation of nonlinear large displacements and moderate rotations of composite beam structures considering a finite element potential energy approach incorporating through the thickness shear strain as an analytical function. This approach was compared to large rotation theories. Test cases were run to evaluate numerical algorithms. Riks method and displacement imposed techniques were employed. The limitations and advantages of both methods were considered. Loading arrangements included concentrated forces as well as moments.

NONLINEAR LARGE DISPLACEMENT AND MODERATE ROTATIONAL CHARACTERISTICS OF COMPOSITE BEAMS INCORPORATING TRANSVERSE SHEAR STRAIN

I. Introduction

Today's aerospace industry has advanced to such a degree that optimum performance is not available from the use of isotropic structural elements. Laminates of composite materials are now used extensively, and they allow optimization of the strength and weight of the structural element. In particular, the sequence, number and orientation of plies in a laminate may be varied to directionally bias the strength of a plate or a shell. Hence, weight may be saved since unneeded or undesirable stiffness is eliminated. Since the laminates are employed in highly optimized conditions (i.e. use as little material as possible to minimize aircraft weight), large factors of safety are not available. Consequently, the optimized structures often approach collapse loads. Thus, the structural engineer must be aware of the loading and displacement configurations that will cause collapse. In addition, knowledge of a structural element's behavior after collapse allows the engineer to design a forgiving structure that avoids catastrophic failure.

In this light, much work has been done to predict the post–collapse behavior of anisotropic plates and shells. One class of problems are those in which the structure becomes geometrically nonlinear but experiences only small strains, so, the material continues to be linearly elastic. In essence, the stiffness of the structure changes not because the material becomes plastic but because the shape of the structure changes radically. To address this class of problems, Palazotto and Dennis have developed a theory and generated a FORTRAN code that traces the equilibrium path of orthotropic cylindrical shells including the following (13):

1. geometric nonlinearity with moderate rotations and large displacements,

2. linear elastic behavior of laminated anisotropic materials,
3. cylindrical shells and flat plates,
4. parabolic transverse distribution of shear stress,
5. all cased in a finite element approach.

This thesis narrows the class of problems considered by Palazotto and Dennis from two dimensions to one dimension. That is, the techniques for plates and shells are simplified to analyze straight beams and arches. All the features of Palazotto and Dennis's theory are retained, only the dimensionality is reduced. The result is a FORTRAN code which is, of course, much more efficient for beams and arches than the two dimensional version. Also, since the one–dimensional case represents the simplest possible case for the theory, the present code allows for uncluttered observation of the rotational and displacement limits of Palazotto and Dennis's theory.

A two–dimensional (2–D) theory to describe the three dimensional (3–D) behavior of thin plates with small deflections was developed by Kirchhoff (20). Kirchhoff assumed that the middle plane remains unstrained, normal strains were small enough to be neglected, and that planes normal to the mid–plane before bending remain normal and unwarped after bending. The third assumption translates to neglecting transverse shear. Love extended Kirchhoff's methods and assumptions to thin shells under small deflections (20). Though these Kirchhoff–Love, or classical, theories are limited to isotropic shells and plates under small deflections, the theories are applicable for many problems and they set the precedent for using the shell's middle surface as a computational datum surface.

Reissner (18) and Mindlin (11) added transverse shear consideration to the classical theory. The Reissner–Mindlin (RM) approach, or first–order transverse shear theory, allows normals to the datum surface to rotate, but not warp, with deformation. Though consideration of transverse shear represents an improvement over classical theory, the RM theory does not satisfy boundary conditions of zero transverse shear stress on the top and bottom shell surfaces. Consequently, finite elements based on first order theory suffer from shear locking as the shell becomes thin. So, shear correction factors are used to eliminate the shear locking effect.

Most recently Reddy (17) and others have presented theories which account for transverse shear by a parabolic shear strain distribution through the plate or shell thickness. This distribution satisfies the zero transverse shear stress boundary conditions on the top and bottom surfaces and, as a result, eliminates shear locking. Reddy also applied this theory to laminated plates (16) where he assumes laminated anisotropy. That is, each ply layer is treated as an orthotropic material. Palazotto and Dennis (13) apply this parabolic transverse shear strain distribution to a nonlinear analysis of cylindrical shells with moderate rotations and large deformations.

Few analytical, closed–form, solutions exist for shell geometries, so, finite elements solutions are almost always used. Since all shell and plate formulations involve simplifications from the three–dimensional case, 3–D elements are generally not used for plate or shell modeling (3). Three types of shell or plate elements are generally used (13): flat elements are fine for plates but many are required to achieve convergence for a shell, two–dimensional or Love theory is used to develop curved shell elements, or curved shell elements are formed by reducing 3–D strain–displacement relations.

Finite elements of the third type above resemble 3–D elements since they have nodes on the top and bottom surfaces. These elements are well suited to first–order shell theory; however, they develop shear locking problems as the shell gets thin. That is, the stiffness due to transverse shear tends to increase very rapidly as the shell becomes thin resulting in a much stiffer mesh. This is a numerical difficulty which can be compensated for by reduced integration (3) or shear correction factors.

Two–dimensional elements had only been applied to linear problems until the recent nonlinear work of Reddy and Palazotto and Dennis.

Roughly paralleling the development of shell and plate theory have been theories dealing with beams and arches (or curved beams). In fact, beam and arch problems are one dimension simpler than plates and shells. However, many structures utilize both isotropic and laminated beams. So, the effort invested in reducing sophisticated 2–D shell and plate theories to 1–D arch and beam theories is well spent since more efficient finite element codes save much time in design and analysis of these simpler problems. Also, in an academic sense, reduction of 2–D theories to 1–D removes coupling between orthogonal directions so that the rotational and displacement limitations of a shell theory can be revealed.

As in shell theory, the three classes of beam theory are classical, or Kirchhoff; first–order transverse shear, or Reissner–Mindlin; and higher–order theories. This introduction considers a slightly different breakdown to discuss theories which, in subsequent chapters, will be compared to results of the present theory. Answers to the following questions will be used to classify different theories.

1. Does the theory include transverse shear? If so, is the shear representation first–order or higher–order?
2. Does the theory allow extensibility (stretching) of the midplane along the beam long axis? If so, are any or all of the higher–order strain terms retained?
3. Does the formulation use total Lagrangian or updated Lagrangian kinematics? Do the kinematics include small angle approximations?

Incidentally, since this research considers geometrically nonlinear problems, all theories considered here are capable of solving geometrically nonlinear problems. Much of the work in nonlinear beam theory consists of studies of circular arches. Arches are popular because they are not only nonlinear, but they display limit points if load is plotted versus displacement for equilibrium states as the arch becomes unstable.

Huddleston (10) does not include transverse shear in his development for deep circular arches. Huddleston's theory does allow for stretching of the midplane; however, higher–order strain terms are not included since he he derives his strain from axial forces and moments through constitutive relations rather than by strain–displacement relations. Finally, Huddleston uses a total Lagrangian approach which is exact in the sense that no approximations are used for trigonometric functions to compute rotation angles. Huddleston's theory is unique among others discussed here in that, rather than use finite elements or finite difference equations, he solves simultaneous, nonlinear, first–order differential equations.

Sabir and Lock (21) also neglect transverse shear in their effort to develop a finite element to handle large deflections of circular arches. This theory does allow the midplane to stretch, though, not all higher–order terms are retained. Sabir and Lock employ total Lagrangian kinematics and their approach uses a unique shape function.

DaDeppo and Schmidt have published numerous articles considering circular arches using various approaches (6, 7, 5). When they do consider transverse shear, DaDeppo and Schmidt allow normals to the midplane to rotate but not warp (5:35). So, their transverse shear theory is first–order and requires shear correction factors. In addition, DaDeppo and Schmidt have dealt with inextensible (6:990) and extensible (5:37) cases. Though, in the latter case, higher–order terms are neglected. DaDeppo and Schmidt use a total Lagrangian approach which is exact (i.e. no trigonometric function approximations) and they use a finite difference approach to reduce the differential equations of equilibrium to algebraic equations.

Epstein and Murray (9) ignore transverse shear in their formulation. However, they do retain all higher order shear terms for midplane extensibility. In addition, they employ exact total Lagrangian kinematics.

Belytschko and Glaum (1) ignore transverse shear and also allow the midplane to stretch. However, not all higher–order strain terms are retained. Of interest in Belytschko and Glaum's work is their use of a corotational, updated Lagrangian, coordinate system in which the coordinate system for each element is updated on each solution increment to account for rigid body motion. Within the corotational ccordinates small angle approximations are used.

More recently, Minguet and Dugundji (12) have developed a theory to predict large deflections of laminated beams. Minguet and Dugundji assume transverse shear strains are constant through–the–thickness and employ an updated Lagrangian elemental coordinate system which represents rigid body motion of the element exactly via Euler angles. The midplane is allowed to stretch but higher–order terms are not included.

The theories discussed above use various techniques to solve geometrically nonlinear problems. The most popular technique for solving the nonlinear algebraic equations that result from discretization of a structure is the Newton–Raphson technique. This iterative technique uses the tangent stiffness at a point on the equilibrium curve to approximate a point further along the curve at a particular load or displacement. The unknown load or displacement at this intermediate point is backed out of the finite element equilibrium equations, a new tangent stiffness is computed, and iteration continues until the requested load or displacement value is achieved within a specified tolerance. A variation is the modified Newton–Raphson technique in which the tangent stiffness is not updated for iterations within an increment. Difficulties arise at limit points where the tangent stiffness matrix becomes singular. Near horizontal limit points, displacement control traverses the singularities since a unique load exists for each displacement. Likewise, near vertical limit points load control passes limit points because unique displacements exist for each load. The logical conclusion is a code capable of switching between load and displacement control to traverse any limit point; Sabir and Lock (21) have devised just such a code.

A more elegant, but more complex, technique has been formulated by Riks (19). Riks adds a constraint equation to the system equations which prescribes a fixed distance from the starting point about which a solution is sought. Incidentally, this fixed distance has come to be referred to in the literature as an "arc length" and Riks' method is often referred to as an "arc length" method. However, the fixed distance prescribed by the constraint equation is not an arc length; it is a radius which describes, in 1–D, an arc along which an intersection with the equilibrium curve is sought. In this work, this distance will be referred to as the search radius. Riks' method has been reformulated for finite elements by Crisfield (4) and Ramm (14).

The present work then, reformulates the 2–D theory originally introduced by Dennis in his dissertation (8) to handle 1–D straight beams and circular arches with geometric nonlinearities and with a capability to accept symmetric laminates. Reduction of 2–D cylindrical shell theory to 1–D first entailed reconsidering the assumptions of Palazotto and Dennis (whose theory will be referred to by their acronym "SLR" for Simplified Large Displacement/Rotation). By making beam–type assumptions major simplifications were made to the constitutive relations and strain–displacement relationships. In addition a new finite element was developed along with a FORTRAN code which incorporates displacement control and Riks method to solve nonlinear problems. Dennis's work has been expanded upon by Smith (22) who added cubic terms to the transverse shear distribution and additional nonlinear terms to midplane stretching strain. Tsai and Palazotto (23) broadened the nonlinear scope of Dennis's code by incorporating Riks technique. This work includes the additional midplane strain terms of Smith, as well as Tsai's adaptation of Riks' technique.

II. Theory

2.1 Constitutive Relations

In reducing the 2-D shell theory of Palazotto and Dennis (13) to 1-D for beams and arches, major simplifications are possible if we first look at the constitutive relations. The stress-strain relations for a single orthotropic ply of a laminate in the ply material coordinates are given by (15)

$$\begin{Bmatrix} \sigma_1' \\ \sigma_2' \\ \sigma_6' \\ \sigma_4' \\ \sigma_5' \end{Bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} & 0 & 0 & 0 \\ Q_{12} & Q_{22} & 0 & 0 & 0 \\ 0 & 0 & Q_{66} & 0 & 0 \\ 0 & 0 & 0 & Q_{44} & 0 \\ 0 & 0 & 0 & 0 & Q_{55} \end{bmatrix} \begin{Bmatrix} \epsilon_1' \\ \epsilon_2' \\ \epsilon_6' \\ \epsilon_4' \\ \epsilon_5' \end{Bmatrix} \tag{2.1}$$

where the contracted notation of table 2.1 is used for primed and unprimed coordinate systems. The Q_{ij} 's are the plane *stress reduced* ply stiffnesses in the material coordinate system, the σ_{ij} 's are the stresses and the ϵ_{ij} 's are the strains. This development assumes that σ_3, or the thru–the–thickness normal stress, may be neglected because a thin shell (or beam or arch) is in an approximate state of plane stress. Where a thin shell is defined as one where the thickness–to–radius of curvature ratio is less than 1/5. The through–the–thickness strain, ϵ_3 is accounted for by constitutive relations, through the Q_{ij}'s, with the in–plane strains ϵ_1 and ϵ_2 as detailed by Palazotto and Dennis (13:35). The Q_{ij}'s may be

Stress		Strain	
contracted	explicit	contracted	explicit
σ_1	σ_{11}	ϵ_1	ϵ_{11}
σ_2	σ_{22}	ϵ_2	ϵ_{22}
σ_3	σ_{33}	ϵ_3	ϵ_{33}
σ_4	σ_{23}	ϵ_4	$2\epsilon_{23}$
σ_5	σ_{13}	ϵ_5	$2\epsilon_{13}$
σ_6	σ_{12}	ϵ_6	$2\epsilon_{12}$

Table 2.1. Contracted Notation Conventions

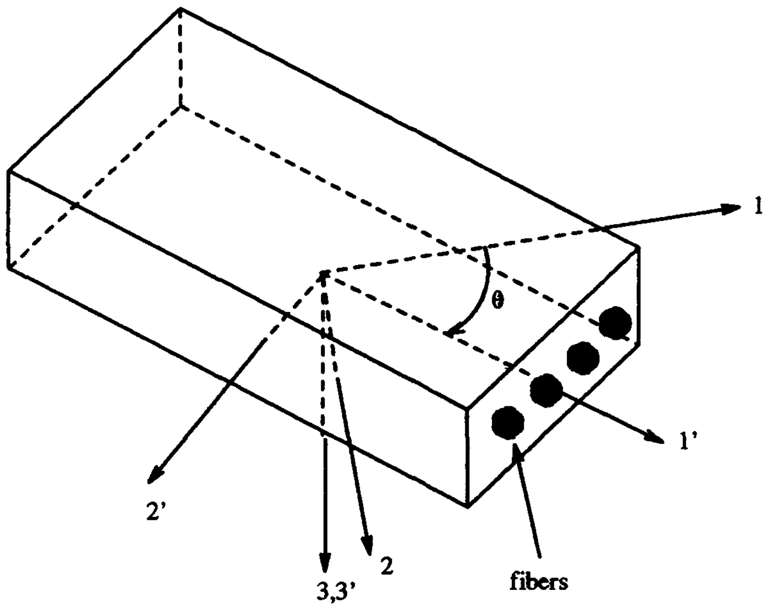

Figure 2.1. Material and Shell Coordinate Systems

expressed in terms of the engineering constants as

$$
\begin{aligned}
Q_{11} &= \frac{E_1}{1-\nu_{12}\nu_{21}}, \quad Q_{12} = \frac{\nu_{12}E_2}{1-\nu_{12}\nu_{21}} = \frac{\nu_{21}E_1}{1-\nu_{12}\nu_{21}} \\
Q_{22} &= \frac{E_2}{1-\nu_{12}\nu_{21}}, \quad Q_{44} = G_{23}, \quad Q_{55} = G_{13}, \quad Q_{66} = G_{12}
\end{aligned}
\tag{2.2}
$$

Here, the E_i 's are Young's moduli, the G_{ij} 's are shear moduli and the ν_{ij} 's are Poisson's ratios.

To be of use in a system of plies, or a laminate, the stiffnesses must be transformed to laminate, or global, coordinates. The transformations of the material Q_{ij}'s to the global $\bar{Q}_{ij}$'s are taken from Smith (22) and are presented in the Appendix. The relationship between material and global coordinates is shown in figure 2.1. Figure 2.2 shows the global coordinates superimposed on an arch. The s coordinate direction is adapted from the 2 direction through scale factors described below. The resulting constitutive relations

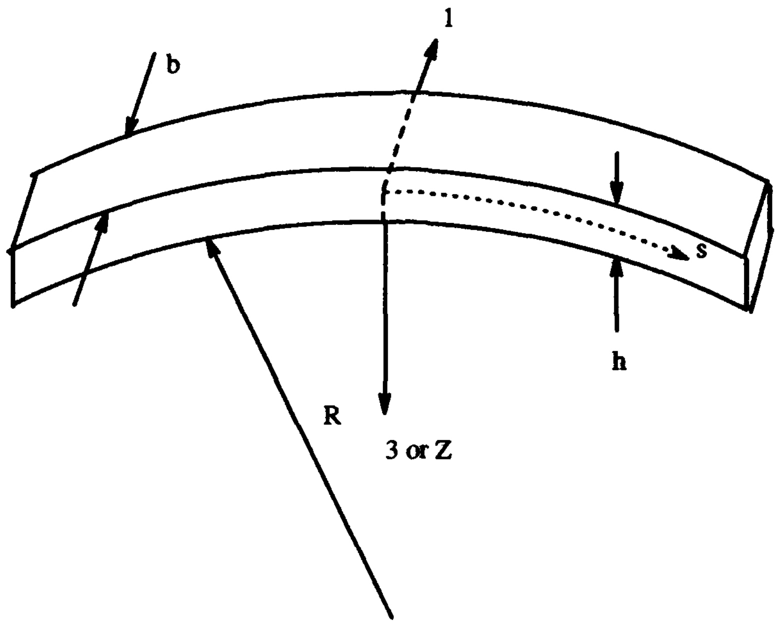

Figure 2.2. Global Coordinate System on Arch Element

for the k^{th} ply in global coordinates are (15)

$$\begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_6 \end{Bmatrix}_k = \begin{bmatrix} \bar{Q}_{11} & \bar{Q}_{12} & \bar{Q}_{16} \\ \bar{Q}_{12} & \bar{Q}_{22} & \bar{Q}_{26} \\ \bar{Q}_{16} & \bar{Q}_{26} & \bar{Q}_{66} \end{bmatrix}_k \begin{Bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_6 \end{Bmatrix}_k \tag{2.3}$$

$$\begin{Bmatrix} \sigma_4 \\ \sigma_5 \end{Bmatrix}_k = \begin{bmatrix} \bar{Q}_{44} & \bar{Q}_{45} \\ \bar{Q}_{45} & \bar{Q}_{55} \end{bmatrix}_k \begin{Bmatrix} \epsilon_4 \\ \epsilon_5 \end{Bmatrix}_k \tag{2.4}$$

Now we simplify these 2–D expressions to 1–D by assuming normal stress in the 1–direction, σ_1, is zero since the width of the beam will be relatively small in that direction and we can neglect stresses due to anticlastic curvature. Also, we'll assume the in–plane shear stress, σ_6, and strain, ϵ_6 are zero since there is no material present on the sides of the beam to exert such shear. For the same reasons we assume $\sigma_5 = \epsilon_5 = 0$. So, our global constitutive relations reduce to

$$\begin{Bmatrix} 0 \\ \sigma_2 \end{Bmatrix}_k = \begin{bmatrix} \bar{Q}_{11} & \bar{Q}_{12} \\ \bar{Q}_{12} & \bar{Q}_{22} \end{bmatrix}_k \begin{Bmatrix} \epsilon_1 \\ \epsilon_2 \end{Bmatrix}_k \tag{2.5}$$

$$\sigma_{4k} = \bar{Q}_{44k}\epsilon_{4k} \tag{2.6}$$

The 1–direction normal strain, ϵ_1, is now dependent on ϵ_2. If we solve for it in terms of ϵ_2, equation 2.5 reduces to

$$\begin{aligned}\sigma_{2k} &= \hat{Q}_{2k}\epsilon_{2k} \\ \hat{Q}_{2k} &= (\bar{Q}_{22} - \frac{\bar{Q}_{12}^2}{\bar{Q}_{11}})_k\end{aligned} \tag{2.7}$$

Where $\hat{Q}_{2k}$ represents a 1–D version of the reduced ply stiffness terms. Incidentally, the 2 direction has been retained as the coordinate along the beam or arch since that's the coordinate Palazotto and Dennis use in the circumferential direction of their cylinder. Hence, equations can be compared easily.

2.2 Strain–Displacement Relations

Palazotto and Dennis (13) represent the strains in a shell by Green's strain tensor such that the physical strains are

$$\epsilon_{ij} = \frac{\gamma_{ij}}{h_i h_j} \tag{2.8}$$

where the γ_{ij} 's are the elements of Green's Strain tensor and the h_i 's are scale factors. Since we've reduced the order of the problem, we need only two components of Green's strain tensor (20).

$$\begin{aligned}\gamma_{22} &= h_2u_{2,2} + \frac{h_2u_3}{h_3}h_{2,3} + \frac{h_2u_1}{h_1}h_{2,1} \\ &+\frac{1}{2}\left(u_{2,2} + \frac{u_3}{h_3}h_{2,3} + \frac{u_1}{h_1}h_{2,1}\right)^2 \\ &+\frac{1}{2}\left(u_{3,2} - \frac{u_2}{h_3}h_{2,3}\right)^2 + \frac{1}{2}\left(u_{1,2} - \frac{u_2}{h_1}h_{2,1}\right)^2 \\ \gamma_{23} &= \frac{1}{2}(h_3u_{3,2} + h_2u_{2,3} - u_2h_{2,3} - u_3h_{3,2}) \\ &+\frac{1}{2}\left(u_{2,3} - \frac{u_3}{h_2}h_{3,2}\right)\left(u_{2,2} + \frac{u_3}{h_3}h_{2,3} + \frac{u_1}{h_1}h_{2,1}\right) \\ &+\frac{1}{2}\left(u_{3,2} - \frac{u_2}{h_3}h_{2,3}\right)\left(u_{3,3} + \frac{u_2}{h_2}h_{3,2} + \frac{u_1}{h_1}h_{3,1}\right) \\ &+\frac{1}{2}\left(u_{1,2} - \frac{u_2}{h_1}h_{2,1}\right)\left(u_{1,3} - \frac{u_2}{h_1}h_{3,1}\right)\end{aligned} \tag{2.9}$$

Where the u_i 's are the global displacements.

For a cylinder and for an arch, where s (in the 2 direction) is the circumferential coordinate, the scale factors reduce to

$$h_1 = h_3 = 1 \quad h_2 = 1 - \tfrac{z}{R} \tag{2.10}$$

So, to find the strains at a point we must first know the displacements. The assumed displacement functions are as shown in equation 2.11. The functions for u_2 and u_3 are the same as those of Palazotto and Dennis for a shell except for lack of dependency on the 1 direction. Also, here, the lateral displacement of the arch or beam, u_1, is zero.

$$\begin{aligned} u_1 = u &= 0 \\ u_2(s,z) &= v\left(1 - \frac{z}{R}\right) + \psi_2 z + \phi_2 z^2 + \gamma_2 z^3 + \theta_2 z^4 \\ u_3(s) &= w \end{aligned} \tag{2.11}$$

Where u , v , and w are displacements of the middle plane of the beam which do not vary through the width, b , of the beam; R is the arch radius of curvature (infinity for a straight beam); ψ_2 is the rotation of normals to the midplane due to bending; and ϕ_2 , γ_2 and θ_2 are coefficients of higher order powers of z , the through–the–thickness coordinate, which are determined by the boundary conditions of zero shear strain on the top and bottom surfaces of the beam. From equations 2.1, 2.8 and 2.10 we have

$$\epsilon_4 = 2\epsilon_{23} = \frac{2\gamma_{23}}{1 - \frac{z}{R}} \tag{2.12}$$

Assuming our beam is thin, in–plane stresses and strains dominate its behavior when compared to the transverse stresses and strains. So, for the transverse strains we include only the linear terms of γ_{23} such that

$$\epsilon_4 = \frac{1}{1 - \frac{z}{R}}\left[u_{3,2} + \left(1 - \frac{z}{R}\right)u_{2,3} + \frac{u_2}{R}\right] \tag{2.13}$$

The values for u_2 and u_3 are now substituted from equation 2.11 and equation 2.13 is evaluated at $z = \pm\frac{h}{2}$ where $\epsilon_4 = 0$. The unknown coefficients are then solved for such that

$$\phi_2 = 0 \qquad \theta_2 = \frac{\gamma_2}{2R} \tag{2.14}$$
$$\left[1 - \frac{h^2}{8R^2}\right]\gamma_2 = -\frac{4}{3h^2}(\psi_2 + w_{,2}) \approx \gamma_2$$

Assuming the beam is thin, i.e. $h/R \leq 1/5$, the term $h^2/8R^2 \leq .005$. So, this term is neglected and the approximation for γ_2 is used. In addition, the fourth order coefficient, θ_2, is neglected since it is, at most, 1/20 the size of the third order term, γ_2. The in plane displacements may now be written as

$$u_2(s,z) = v(1 - z/R) + z\psi_2 + z^3k(\psi_2 + w_{,2}) \tag{2.15}$$

where $k = -\frac{4}{3h^2}$. Next, this reduced version of u_2 is substituted into 2.13 along with u_3 resulting in

$$\epsilon_4 = \frac{1}{1 - z/R}(w_{,2} + \psi_2)\left[1 - 4z^2/h^2 + \frac{8z^3}{3h^2R}\right] \tag{2.16}$$

Again, considering order of magnitudes for $h/R \leq 1/5$, the final term is less than 1/15 the size of the next largest; so, it is neglected. Finally, we have

$$\epsilon_4 = \frac{1}{1 - z/R}(w_{,2} + \psi_2)\left[1 - 4z^2/h^2\right] \tag{2.17}$$

Equation 2.17 explicitly shows the slope of the elastic curve, $w_{,2}$, and rotation due to bending of sections normal to the elastic curve, ψ_2. Figure 2.3 illustrates the sign convention employed in equation 2.15 for ψ_2 and $w_{,2}$ and for rotations due to transverse shear β_2. In figure 2.3, the quantities ψ_2 and β_2 are shown as rotations about a particular point and $w_{,2}$ is shown as the slope of the tangent to the elastic curve at that point. In both illustrations, $w_{,2}$ is shown in a positive sense since w increases downward and to the right with respect to the 2 axis. In these views, ψ_2 and β_2 are positive in the counterclockwise direction since rotations in this direction would cause points on a perpendicular to the 2-axis in the z-direction to move to the right, which is a positive u_2. In the top drawing, ψ_2 and β_2 are both negative. This sign convention is reflected in equation 2.15 where a minus ψ_2 would produce motion in a minus 2 direction. In the bottom drawing of figure 2.3, ψ_2 is again negative, however, in this case it is larger in magnitude than the positive $w_{,2}$; so, β_2 must be positive to resolve the bending and shear rotational quantities to the physical state of the beam, represented by $w_{,2}$. Mathematically, the relationship between these three quantities may be expressed as

$$|w_{,2} + \psi_2| = -\beta_2 \tag{2.18}$$

Also figure 2.4 shows, for an initially rectangular beam section, the relation between the tangent to the elastic curve, $w_{,2}$, and the rotational quantities for, first, zero transverse shear and, second, for pure transverse shear.

For derivation of the strain–displacement relationship considering the in–plane normal strains, we return to equation 2.8 such that

$$\epsilon_2 = \frac{\gamma_{22}}{h_2^2} \tag{2.19}$$

In this case, however, we retain all the terms for γ_{22} from equation 2.9 since these in–plane strains and the resulting stresses dominate the behavior of a thin beam. Palazotto and Dennis (13), on the other hand, eliminate thirteen higher order terms for their cylindrical shell representation. These terms, however, were included by Smith (22) in his later work. Since we've already made major simplifications and for the sake of completeness, these terms are retained here. The in–plane strains are represented by defining terms ϵ_2^0 and χ_{2p} that are not a function of the thickness coordinate z, such that

$$\epsilon_2 = \epsilon_2^0 + \sum_{p=1}^{7} z^p \chi_{2p} \tag{2.20}$$

where the $1/h_2^2$ term is represented by the truncated binomial expansion

$$\frac{1}{(1 - z/R)(1 - z/R)} = 1 + \frac{2z}{R} + \cdots \tag{2.21}$$

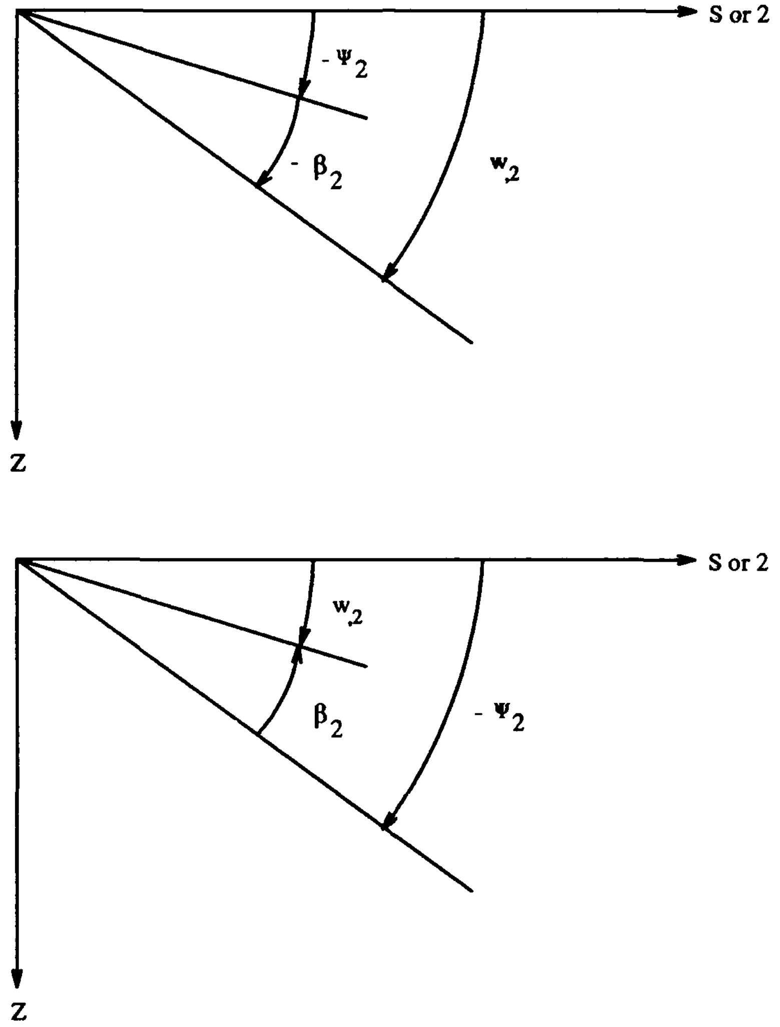

Figure 2.3. Relationships between bending, ψ_2, slope, $w_{,2}$, and shear,β

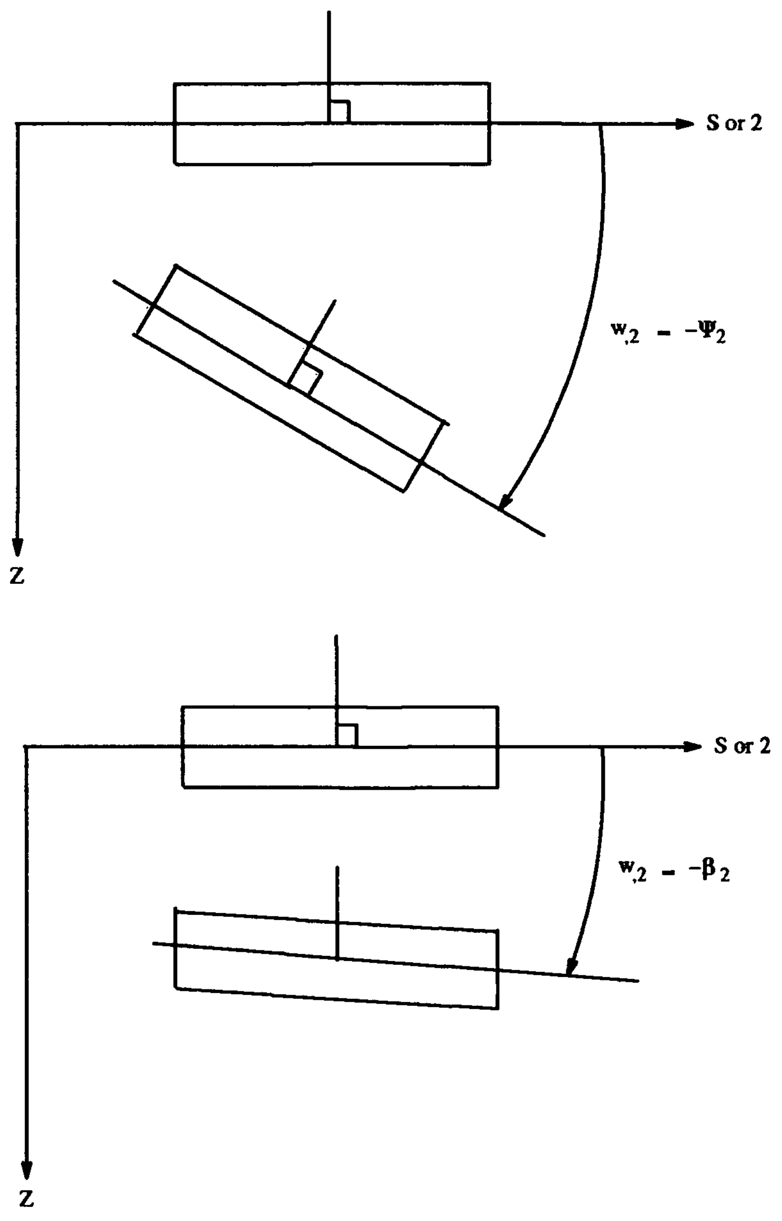

Figure 2.4. Shear, Bending and Slope

and

$$
\begin{aligned}
\epsilon_2^0 &= v_{,2} - wc + 1/2\left(v_{,2}^2 + w_{,2}^2 + v^2c^2 + w^2c^2\right) + vw_{,2}c - v_{,2}wc \\
\chi_{21} &= \psi_{2,2} - wc^2 + w_{,2}^2c + w^2c^3 - c^2\left(v_{,2}w - vw_{,2}\right) + v\psi_2c^2 + v_{,2}\psi_{2,2} - c\left(\psi_{2,2}w - \psi_2w_{,2}\right) \\
\chi_{22} &= \psi_{2,2}c + \frac{1}{2}\left(\psi_{2,2}^2 + \psi_2^2c^2\right) + v\psi_2c^3 - 2c^2\left(\psi_{2,2}w - \psi_2w_{,2}\right) + \psi_{2,2}v_{,2}c \\
&\quad \underline{-\frac{3}{2}(c^4v^2 + c^2v_{,2}^2) - c^2v_{,2} + 2c^3(v_{,2}w - vw_{,2})} \\
\chi_{23} &= k\left(w_{,22} + \psi_{2,2}\right) + c\psi_{2,2}^2 + \psi_2^2c^3 + kv_{,2}\left(w_{,22} + \psi_{2,2}\right) \\
&\quad +vkc^2\left(w_{,2} + \psi_2\right) - wkc\left(w_{,22} + \psi_{2,2}\right) + w_{,2}kc\left(w_{,2} + \psi_2\right) \qquad (2.22) \\
&\quad \underline{+c^5v^2 + c^3v_{,2}^2 - 2c^2(c^2v\psi_2 + v_{,2}\psi_{2,2})} \\
\chi_{24} &= kc\left(w_{,22} + \psi_{2,2}\right) + vkc^3\left(w_{,2} + \psi_2\right) + 2kc^2\left(-ww_{,22} - w\psi_{2,2} + w_{,2}^2 + w_{,2}\psi_2\right) \\
&\quad +k\psi_{2,2}\left(w_{,22} + \psi_{2,2}\right) + \psi_2kc^2\left(w_{,2} + \psi_2\right) + v_{,2}kc\left(w_{,22} + \psi_{2,2}\right) \\
\chi_{25} &= 2kc\left[\psi_{2,2}\left(w_{,22} + \psi_{2,2}\right) + \psi_2c^2\left(w_{,2} + \psi_2\right)\underline{-c(v_{,2}w_{,22} + v_{,2}\psi_{2,2}) - c^3(vw_{,2} + v\psi_2)}\right] \\
\chi_{26} &= \frac{k^2}{2}\left[w_{,22}^2 + 2w_{,22}\psi_{2,2} + \psi_{2,2}^2 + c^2\left(w_{,2}^2 + 2w_{,2}\psi_2 + \psi_2^2\right)\right] \\
\chi_{27} &= k^2c\left[\left(w_{,22} + \psi_{2,2}\right)^2 + c^2\left(w_{,2} + \psi_2\right)^2\right]
\end{aligned}
$$

where $c = 1/R$ and the terms neglected in the prior work are underlined.

At this point, it is useful to consider how others handle axial strain. Schmidt and DaDeppo (6), for instance, represent extensional strain, ϵ of the centroidal curve of an arch rib as

$$2\epsilon + \epsilon^2 = 2\kappa(u'\cos\phi + v'\sin\phi) + \kappa^2(u'^2 + v'^2) \qquad (2.23)$$

where κ is the arch curvature, u and v are horizontal and vertical displacement components of a point on the centroidal curve, and prime denotes derivative with respect to the angle ϕ formed by a normal to the undeformed centroidal curve and a vertical reference line. Schmidt and DaDeppo further define a quantity β, the angle of rotation of a tangent to the centroidal curve, which is comparable to our slope $w_{,2}$.

$$\beta = \arctan\frac{\kappa(-u'\sin\phi + v'\cos\phi)}{1 + (u'\cos\phi + v'\sin\phi)} \qquad (2.24)$$

The most obvious difference between our ϵ_2^0 and Schmidt and DaDeppo's ϵ is their use of trigonometric functions to define angles. This means that Schmidt and DaDeppo can more accurately represent large rotations than the present theory. In fact, comparisons will be made between the two theories for displacements of a very deep arch and the rotational limits of the present theory will be quantified.

For ϵ_4, the shape factor term is truncated after the constant term such that

$$\frac{1}{1-z/R} = 1 + z/R + \frac{z^2}{R^2(1-z/R)} \approx 1 \tag{2.25}$$

since for a shell where $h/R \leq 1/5$ and z never exceeds $h/2$ since it is measured downward from the midplane, the second term is at most 10% of the first term. So, we have

$$\epsilon_4 = \epsilon_4^0 + z^2\chi_{42} \tag{2.26}$$

where $\epsilon_4^0 = w_{,2} + \psi_2$ is the midplane shear strain, $\chi_{42} = k\epsilon_4^0$ is not a function of thickness and $k = -\frac{4}{3h^2}$ is a thickness parameter.

2.3 Beam Potential Energy

In general, for an elastic system the potential energy, Π_p is

$$\Pi_p = U + V \tag{2.27}$$

where U is the internal strain energy and V is the work done by external forces. The internal strain energy consists of a strain energy density function, W^*, integrated over the volume where, for a conservative system with small strains,

$$W^* = \frac{1}{2}a_{ijkl}\epsilon_{ij}\epsilon_{kl} \tag{2.28}$$

where the a_{ijkl} represents a constitutive matrix. So,

$$U = \int_V W^* dV \tag{2.29}$$

where dV is an infinitesimal volume element. In our simplified case, the strain energy reduces to

$$U = U_1 + U_2 \tag{2.30}$$

where,

$$\begin{aligned} U_1 &= \frac{1}{2} b \int_l \int_h \hat{Q}_{2k} \epsilon_2^2 dz ds \\ U_2 &= \frac{1}{2} b \int_l \int_h Q_{44k} \epsilon_4^2 dz ds \end{aligned} \tag{2.31}$$

Here, b is the beam or arch width, l is the beam or arch length and h is the beam or arch height. Note that all the terms in the energy expressions are scalars. We previously derived ϵ_2 as equation 2.20, so we have

$$\epsilon_2^2 = \left(\epsilon_2^0\right)^2 + 2\left(\epsilon_2^0\right) \chi_{2p} z^p + \left(\chi_{2p} z^p\right)^2 \tag{2.32}$$

where p =1,2,3,4,5,6,7. This expression allows us to redefine our U_1 integral as

$$U_1 = \frac{1}{2} b \int_l \left(\mu_1 + \mu_2 + \mu_3\right) ds \tag{2.33}$$

where,

$$\begin{aligned} \mu_1 &= \int_h \hat{Q}_{2k} (\epsilon_2^0)^2 dz = (\epsilon_2^0)^2 A \\ \mu_2 &= \int_h 2\hat{Q}_{2k} \epsilon_2^0 \chi_{2p} z^p dz = 2\epsilon_2^0 \left(\chi_{21} B + \chi_{22} D + \chi_{23} E + \chi_{24} F + \chi_{25} G\right) \\ &+ 2\epsilon_2^0 \left(\chi_{26} H + \chi_{27} I\right) \\ \mu_3 &= \int_h \hat{Q}_{2k} (\chi_{2p} z^p)^2 dz = \chi_{21}\chi_{21} D + 2\chi_{21}\chi_{22} E + (2\chi_{21}\chi_{23} + \chi_{22}\chi_{22}) F \\ &+ 2(\chi_{21}\chi_{24} + \chi_{22}\chi_{23}) G + (2\chi_{21}\chi_{25} + 2\chi_{22}\chi_{24}) H + 2(\chi_{21}\chi_{26} + \chi_{22}\chi_{25} + \chi_{23}\chi_{24}) I \\ &+ (2\chi_{21}\chi_{27} + 2\chi_{22}\chi_{26} + 2\chi_{23}\chi_{25} + \chi_{24}\chi_{24}) J + 2(\chi_{22}\chi_{27} + \chi_{23}\chi_{26} + \chi_{24}\chi_{25}) K \\ &+ (2\chi_{23}\chi_{27} + 2\chi_{24}\chi_{26} + \chi_{25}\chi_{25}) L + 2(\chi_{24}\chi_{27} + \chi_{25}\chi_{26}) P + (2\chi_{25}\chi_{27} + \chi_{26}\chi_{26}) R \\ &+ 2\chi_{26}\chi_{27} S + \chi_{27}\chi_{27} T \end{aligned} \tag{2.34}$$

where the μ_i are strain energy "packets" that have been integrated through the thickness dimension. The elasticity terms ($A, B, D, E, F, G, H, I, J, K, L, P, R, S, T$) in equation 2.34 are also integrated through the thickness and are defined as

$$[A, B, D, E, F, G, H, I, J, K, L, P, R, S, T] = \int_h \hat{Q}_{2k} \left[1, z, z^2, z^3, z^4, z^5, z^6, z^7, z^8, z^9, z^{10}, z^{11}, z^{12}, z^{13}, z^{14}\right] dz \qquad (2.35)$$

Where $\hat{Q}_2$ will vary through–the–thickness for an anisotropic laminate. For a laminate that is symmetric about the midplane the elasticity terms that multiply odd powers of z (i.e. B, E, G,...,S) are zero. The present analysis assumes symmetric laminates.

For the shear strain energy from equation 2.31 we have

$$U_2 = \frac{1}{2} b \int_l \int_h \left(\epsilon_4^0 + \chi_{42} z^2\right)^2 Q_{44k} dz ds \qquad (2.36)$$

which is evaluated similarly to the above to yield

$$U_2 = \frac{1}{2} b \int_l \left[\left(\epsilon_4^0\right)^2 AS + 2\left(\epsilon_4^0\right) \chi_{42} DS + \left(\chi_{42}\right)^2 FS\right] ds = \frac{1}{2} b \int_l \mu_s ds \qquad (2.37)$$

where AS, DS, FS are defined by equation 2.35 if Q_{44} is substituted for $\hat{Q}_2$ for the constant, second and fourth power z terms and μ_s represents a "packet" of shear strain energy integrated through the thickness.

2.4 Finite Element Solution

A finite element solution is advantageous for our problem since taking the first variation of equation 2.27 results in nonlinear differential equations for a continuous beam. If we select discrete nodes on the beam, though, to compute our displacements, the equations become nonlinear algebraic equations. Portions between the nodes are our finite elements and we account for their stiffness through our volumetric definition of the strain energy. Displacements within the elements are computed via the nodal displacements and interpolation functions. The nonlinearity of the simultaneous algebraic equations is removed by using an incremental/iterative approach to achieve equilibrium by varying the load or the displacement. The development presented here is an overview of that presented by Palazotto and Dennis (13). Equation 2.27 can be rewritten as

$$\Pi_p = \frac{q^T}{2}\left[K + \frac{N_1(q)}{3} + \frac{N_2(q^2)}{6}\right] q - q^T R \tag{2.38}$$

where q is a column vector of displacements at the nodes, the bracketed expression is a form of the tangent stiffness matrix and R is a column vector of loads at the nodes. Note that the N_1 term in our tangent stiffness expression is a linear function of q and that N_2 is quadratic in q. The K matrix contains constant stiffness terms. The last term in equation 2.38 is negative since work done by external forces represents a *loss* of potential energy to the system. If we take the first variation of the potential energy and set it equal to zero we have a statement of the virtual work principal for an equilibrium configuration of the system

$$\delta\Pi_p = \delta q^T \left\{\left[K + \frac{N_1}{2} + \frac{N_2}{3}\right] q - R\right\} = 0 \tag{2.39}$$

where the tangent stiffness matrix form is altered because it actually contains linear and quadratic q terms. If we call the braced expression in equation 2.39 $F(q)$, for an arbitrary displacement, q, we must have

$$F(q) = 0 \tag{2.40}$$

where $F(q)$ represents nonlinear algebraic equations in q. These equations are linearized by adding a small increment Δq and writing a truncated Taylor series expansion for $F(q)$.

$$F(q + \Delta q) = F(q) + \frac{\partial F}{\partial q}\Delta q + \cdots = 0 \tag{2.41}$$

or,

$$\frac{\partial F}{\partial q}\Delta q = -F(q) \tag{2.42}$$

Expanding the partial differentiation, we obtain

$$\frac{\partial F}{\partial q} = \frac{\partial}{\partial q}\left\{\left[K + \frac{N_1}{2} + \frac{N_2}{3}\right] q - R\right\} = K + N_1 + N_2 \tag{2.43}$$

Substituting, we have

$$[K + N_1 + N_2]\Delta q = -\left[K + \frac{N_1}{2} + \frac{N_2}{3}\right] q + R \tag{2.44}$$

where

$$[K + N_1 + N_2] = K_T \tag{2.45}$$

K contains constant stiffness terms, N_1 contains stiffness coefficients linear in displacement and N_2 contains terms quadratic in displacement. Various iteration/incrementation schemes may be employed to solve equation 2.44; they will be discussed in a later section. Now, since each term in K_T , the tangent stiffness matrix, is a 7x7 matrix which is, in turn, a product of vector and matrix multiplication, the form of each term can vary through the variation and Taylor series expansion processes. Dennis (8) presents an extensive development in this regard where substitutions are presented for terms in K, N_1 and N_2 to minimize manipulation of arrays in actual computations.

Recalling from equations 2.31, 2.33 and 2.38 that

$$U = \frac{q^T}{2}\left[K + \frac{N_1}{3} + \frac{N_2}{6}\right] q = \frac{1}{2} b \int_l (\mu_1 + \mu_2 + \mu_3 + \mu_s) ds \tag{2.46}$$

we seek stiffness terms such that

$$U = \frac{1}{2} b \int_l d^T \left[\hat{K} + \frac{\hat{N}_1}{3} + \frac{\hat{N}_2}{6}\right] d\, ds \tag{2.47}$$

where $\hat{K}$, $\hat{N}_1$ and $\hat{N}_2$ are stiffness coefficients as explained above but yet to be adapted to a particular element and d is the displacement gradient vector,

$$d^T = \{v \;\; v_{,2} \;\; w \;\; w_{,2} \;\; w_{,22} \;\; \psi_2 \;\; \psi_{2,2}\} \tag{2.48}$$

The displacement terms in d are exactly those required for our strain–displacement relations in equations 2.22 and 2.26.

As shown previously, the μ_i expressions are functions of products of the strains which are, in turn, products of terms of the displacement gradient vector, d. So, we break the strains up into linear and nonlinear functions of d. Starting with ϵ_2 as represented by equations 2.20 and 2.22

$$\begin{aligned} \epsilon_2^0 &= L_0^T d + \frac{1}{2} d^T H_0 d \\ \chi_{2p} &= L_p^T d + \frac{1}{2} d^T H_p d \end{aligned} \tag{2.49}$$

where p =1,2,3,4,5,6,7, the L_i 's are column vectors and the H_i 's are symmetric matrices. The values in the L_i's and H_i's are constants and are presented in the Appendix. To illustrate use of the L_i vectors and H_i matrices, consider the ϵ_2^0 term of equations 2.22. With all terms of L_0, H_0 and d shown, this appears as

$$\begin{aligned} \epsilon_2^0 &= \{0 \;\; 1 \;\; -c \;\; 0 \;\; 0 \;\; 0 \;\; 0\} \begin{Bmatrix} v \\ v_{,2} \\ w \\ w_{,2} \\ w_{,22} \\ \psi_2 \\ \psi_{2,2} \end{Bmatrix} \\ &+ \frac{1}{2}\{v \;\; v_{,2} \;\; w \;\; w_{,2} \;\; w_{,22} \;\; \psi_2 \;\; \psi_{2,2}\} \begin{bmatrix} c^2 & 0 & 0 & c & 0 & 0 & 0 \\ 0 & 1 & -c & 0 & 0 & 0 & 0 \\ 0 & -c & c^2 & 0 & 0 & 0 & 0 \\ c & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} v \\ v_{,2} \\ w \\ w_{,2} \\ w_{,22} \\ \psi_2 \\ \psi_{2,2} \end{Bmatrix} \end{aligned} \tag{2.50}$$

Similarly, for ϵ_4,

$$\epsilon_4^0 = S_0^T d \qquad \chi_{42} = S_4^T d \tag{2.51}$$

where the S_i 's are column vectors also presented in the Appendix.

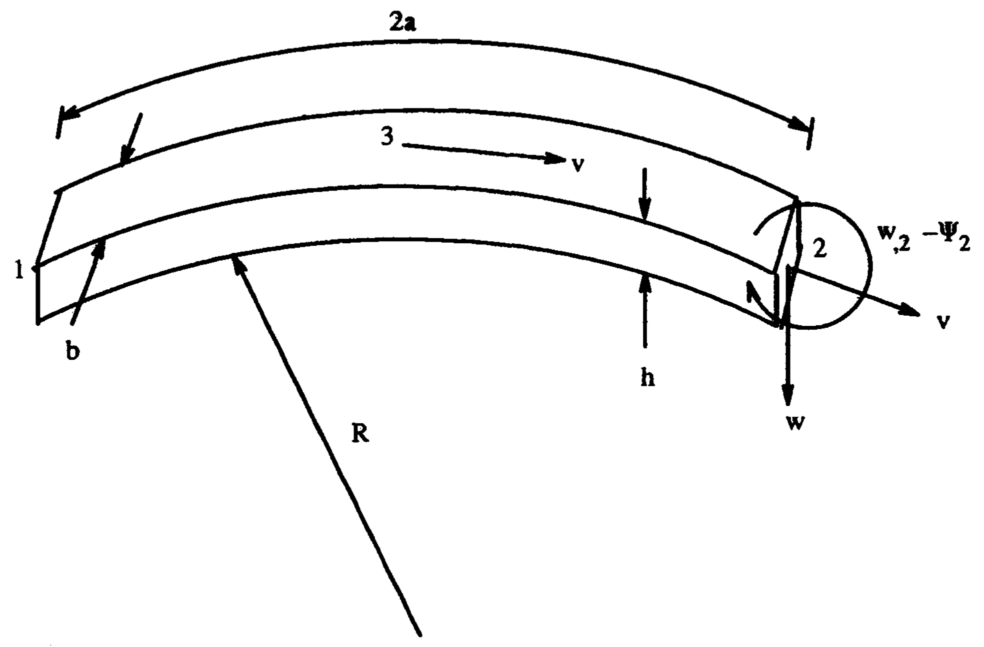

Figure 2.5. Beam Finite Element

The $\hat{K}$, $\hat{N}_1$ and $\hat{N}_2$ matrices are formed through the definition of the μ_i's from equations 2.34 and 2.37 and are presented in the Appendix. Finally, the stiffness arrays are adapted to a finite element through shape functions such that

$$K = b\int_l \mathcal{D}^T\hat{K}\mathcal{D}ds \quad N_1 = b\int_l \mathcal{D}^T\hat{N}_1\mathcal{D}ds \quad N_2 = b\int_l \mathcal{D}^T\hat{N}_2\mathcal{D}ds \tag{2.52}$$

where $\mathcal{D}$ is an array of shape functions and their derivatives as described below. These are the stiffness arrays that are substituted into equations 2.45 and 2.44 for solution.

Now, we adapt this element independent representation to a one–dimensional beam –type finite element as shown in figure 2.5. Note that we've included a middle node with only one degree of freedom v. This is an attempt to capture all energy dissipated through membrane extension due to beam bending. Initially, this theory omitted this middle v node and all results were stiffer than solutions obtained on Palazotto and Dennis's SLR program. This additional degree of freedom allows a more exact representation of the membrane stretching and results in a more flexible solution.

Since we have $w_{,2}$, the derivative of w with respect to the beam axis, defined at the end nodes, we need continuity of this derivative between the elements. So, we use Hermitian shape functions to achieve C^1 continuity for w (3). The other displacement variables v and ψ_2 require only C^0 continuity. Two nodal values for ψ_2 require linear interpolation so Lagrangian shape functions are used. The three nodal values for v, however, require quadratic interpolation functions.

If D is an array of Lagrangian shape functions, N_i, quadratic shape functions, Q_i, and Hermitian shape functions, H_{ij}, and their derivatives with respect to *natural* coordinate $\eta = s/a$ (where $2a$ is the length of the finite element), we have the following relationship between the displacement gradient vector, d, and the nodal variables.

$$d(\eta) = Dq = \begin{bmatrix} Q_1 & 0 & 0 & 0 & Q_3 & Q_2 & 0 & 0 & 0 \\ Q_{1,\eta} & 0 & 0 & 0 & Q_{3,\eta} & Q_{2,\eta} & 0 & 0 & 0 \\ 0 & 0 & H_{11} & H_{12} & 0 & 0 & 0 & H_{21} & H_{22} \\ 0 & 0 & H_{11,\eta} & H_{12,\eta} & 0 & 0 & 0 & H_{21,\eta} & H_{22,\eta} \\ 0 & 0 & H_{11,\eta\eta} & H_{12,\eta\eta} & 0 & 0 & 0 & H_{21,\eta\eta} & H_{22,\eta\eta} \\ 0 & N_1 & 0 & 0 & 0 & 0 & N_2 & 0 & 0 \\ 0 & N_{1,\eta} & 0 & 0 & 0 & 0 & N_{2,\eta} & 0 & 0 \end{bmatrix} \begin{Bmatrix} v^{(1)} \\ \psi_2^{(1)} \\ w^{(1)} \\ w_{,2}^{(1)} \\ v^{(3)} \\ v^{(2)} \\ \psi_2^{(2)} \\ w^{(2)} \\ w_{,2}^{(2)} \end{Bmatrix} \tag{2.53}$$

where

$$d(\eta)^T = \{v \;\; v_{,\eta} \;\; w \;\; w_{,\eta} \;\; w_{,\eta\eta} \;\; \psi_2 \;\; \psi_{2,\eta}\} \tag{2.54}$$

and the (i) postscripts in equation 2.53 are local element node numbers.

The shape functions were derived according to (3) and are as follows.

$$\begin{aligned} N_1 &= \frac{1}{2}(1-\eta) & N_2 &= \tfrac{1}{2}(1+\eta) \\ H_{11} &= \frac{1}{4}(2-3\eta+\eta^3) & H_{12} &= \tfrac{a}{4}(1-\eta-\eta^2+\eta^3) \\ H_{21} &= \frac{1}{4}(2+3\eta-\eta^3) & H_{22} &= \tfrac{a}{4}(-1-\eta+\eta^2+\eta^3) \end{aligned} \tag{2.55}$$

$$Q_1 = \frac{1}{2}(\eta^2 - \eta) \quad Q_2 = \tfrac{1}{2}(\eta^2 + \eta) \quad Q_3 = 1 - \eta^2 \tag{2.56}$$

Finally, the derivatives in the displacement gradient vector, d, are transformed to global coordinates by the inverse of the Jacobian matrix , J ,($\Gamma = J^{-1}$) where

$$d(s) = \Gamma d(\eta) = \Gamma D q \tag{2.57}$$

and

$$\Gamma = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/a & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/a & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/a^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/a \end{bmatrix} \tag{2.58}$$

So, the final shape function array to be used in equation 2.52 for the stiffness arrays is $\mathcal{D} = \Gamma D$.

2.5 Numerical Solution Algorithms

All stiffness terms are now defined and we can rewrite equation 2.44 as follows

$$\sum_{k=1}^{n} \left[\int_l \mathcal{D}^T \left[\hat{K} + \hat{N}_1 + \hat{N}_2 \right] \mathcal{D} ds \right]_k \Delta q = - \sum_{k=1}^{n} \left[\int_l \mathcal{D}^T \left[\hat{K} + \frac{\hat{N}_1}{2} + \frac{\hat{N}_2}{3} \right] \mathcal{D} ds \right]_k q + R \tag{2.59}$$

where n is the number of finite elements in the beam or arch and the integration is over the length of each element, k. R is the global load array with as many rows as total degrees of freedom and q and Δq are global arrays for displacement assembled from elemental displacements.

Gauss quadrature is used to numerically evaluate the integrals in equation 2.59. Taking the integral on the left-hand side as an example we have

$$\int_l \mathcal{D}^T \left[\hat{K} + \hat{N}_1 + \hat{N}_2 \right] \mathcal{D} ds \;=\; \int_{-1}^{1} \mathcal{D}^T \left[\hat{K} + \hat{N}_1 + \hat{N}_2 \right] \mathcal{D} \, detJ \, d\eta \tag{2.60}$$

$$= \sum_{i=1}^{m} w_i \phi(\eta_i)$$

where

$detJ$ = determinant of the Jacobian matrix

$\phi(\eta_i) = \mathcal{D}^T \left[\hat{K} + \hat{N}_1 + \hat{N}_2\right] \mathcal{D} det\ J$, evaluated at the Gauss points, η_i

w_i = a weighting factor.

The range of i depends on the order of quadrature. According to (3), order m quadrature is required to exactly evaluate an integral with polynomials of degree $2m - 1$. For linear solutions (discussed below) fourth order, or 4-point, quadrature gives exact stiffness integration. This is because linear solutions include only the constant stiffness coefficient matrix, $\hat{K}$. Accordingly, the integrand is a 6-degree polynomial because $\mathcal{D}$ and $\mathcal{D}^T$ include cubic polynomials. On the other hand, nonlinear solutions require 7-point quadrature since the $\hat{N}_2$ term introduces a degree 6 polynomial resulting in a 12-degree polynomial for the integrand. In practice, experience with the code based on this work has shown that 5-point quadrature gives virtually the same results as 7-point quadrature. Five-point quadrature was used for all problems.

In the Shell code, Dennis (8) includes a solution algorithm for linear problems. This allows comparison with linear solutions from other work to verify the code and also provides the basis for the first iteration of the nonlinear solution. For a linear solution equation 2.59 becomes

$$\sum_{k=1}^{m} \left[\int_{-1}^{1} \mathcal{D}^T \hat{K} \mathcal{D}\ detJ\ d\eta\right]_k q = R \tag{2.61}$$

where the integral is evaluated numerically. Dennis employs an elimination scheme to zero-out all terms, save for a 1 on the diagonal, of the rows and columns of the global stiffness array for prescribed degrees of freedom. Solution of the simultaneous equations is carried out by Gaussian elimination. The resulting global displacement vector is used along with the strain-displacement and constitutive relationships previously developed to evaluate stresses at the Gauss points.

For nonlinear problems two Newton–Raphson approaches are employed. The first, displacement control, is employed by Dennis (8) in the Shell code. The second, a Riks–Wempner algorithm developed by Tsai and Palazotto (23), uses prescribed arc–lengths to search for the equilibrium path. These schemes are necessary to traverse the limit points inherent to unstable structures.

Consider the equilibrium path shown in figure 2.6. The limit point at A is a snap–through point where the structure instantaneously sheds load and snaps to an inverted configuration at C where it can again support increasing load. At the limit point A the tangent stiffness matrix is singular. However, if we prescribe displacements which traverse this limit point, unique solutions are available for each displacement. This is the advantage of the displacement control technique. At limit points such as B in the figure, on the other hand, displacement control breaks down since more than one solution exists for a given displacement. Points like B are called snap–back points and can be traversed by load–control or approaches that search along an arc–length to find the correct equilibrium path. The displacement control algorithm used here is as presented by Dennis (8) and the incremental equation follows.

$$[K + N_1(q_{r-1}) + N_2(q_{r-1}^2)]\,\Delta q_{r-1} = -\left[K + \frac{N_1(q_{r-1})}{2} + \frac{N_2(q_{r-1}^2)}{3}\right] q_{r-1} + R_1 \qquad (2.62)$$

Where r represents an iteration number within a prescribed displacement increment. For the first iteration of the first increment only the constant K matrix is employed. This linear system is solved for the displacement vector which, in turn, is used to compute, N_1 and N_2 for the next iteration. It is important to realize, for displacement control, only one degree of freedom is normally fixed and represents a constraint on the system of equations. So, $n - 1$ terms in the *vector* q are updated at each iteration within a displacement increment so the tangent stiffness matrix can also be updated at each iteration. Iteration continues until convergence for that increment is achieved. Convergence is based on the following formula.

$$\frac{\sqrt{\sum_i (q_r^i)^2} - \sqrt{\sum_i (q_{r-1}^i)^2}}{\sqrt{\sum_i (q_1^i)^2}} \times 100\% \leq TOL \qquad (2.63)$$

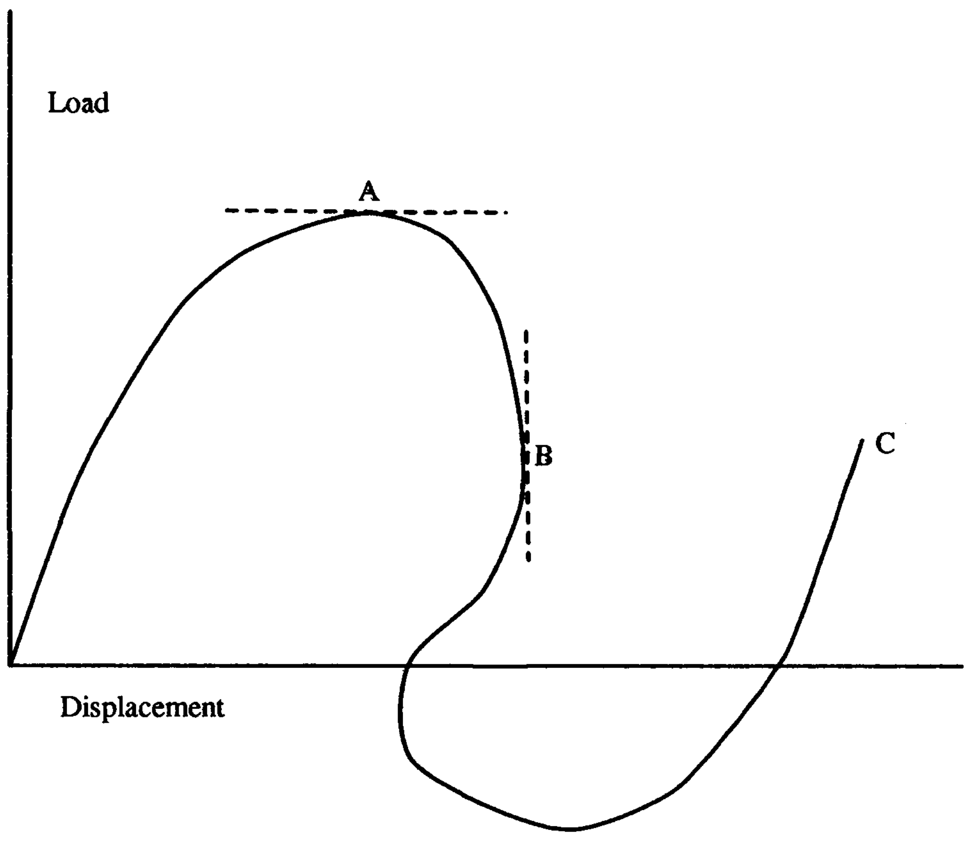

Figure 2.6. Generic Equilibrium Curve

where q_r^i, q_{r-1}^i and q_1^i are the elements of q for the rth, $(r-1)$th and 1st iterations of the increment; i is summed over the total number of degrees–of–freedom and TOL is a user defined tolerance. Typical values of TOL are 0.01% for displacement control and as low as 0.001% for the Riks method.

We now consider the Riks method to trace equilibrium through points like B in figure. As implemented here, the Riks method follows Crisfield's (4) development and the subsequent adaptation to FORTRAN code closely follows the development of Tsai and Palazotto (23).

To develop the Riks algorithm for our case, we introduce a scalar loading parameter, λ , into our equilibrium equation such that

$$F(q,\lambda) = \left(K + \frac{N_1}{2} + \frac{N_2}{3}\right) q - \lambda R = 0 \tag{2.64}$$

Now consider an iterative version of the same equation for iterations i and $i+1$.

$$F(q_{i+1},\lambda_{i+1}) = F(q_i,\lambda_i) + \frac{\partial F}{\partial q}\delta q_i + \frac{\partial F}{\partial \lambda}\delta\lambda_i = 0 \tag{2.65}$$

or

$$\frac{\partial F}{\partial q}\delta q_i + \frac{\partial F}{\partial \lambda}\delta\lambda_i = -F(q_i,\lambda_i) \tag{2.66}$$

Substituting $\frac{\partial F}{\partial q} = K + N_1 + N_2 = K_T$ and $\frac{\partial F}{\partial \lambda} = -R$ we have

$$K_T\delta q_i = \delta\lambda_i R - F(q_i,\lambda_i) \tag{2.67}$$

Now, since we've introduced another unknown, λ, to our system of n equations (where n = number of degrees of freedom in our finite element model), we need one more equation. This additional equation is a constraint equation which establishes a constant radius of length Δl to, in turn, establish an arc along which we search for equilibrium. Geometrically the constraint equation amounts to the Pythagorean theorem where

$$\Delta q_{i+1}^T \Delta q_{i+1} + \Delta\lambda_{i+1}^2 R^T R = \Delta l^2 \tag{2.68}$$

At this point, it's important to remember the purpose of any numerical technique for solving nonlinear equations. The purpose is to establish a path along which successive iterations get closer and closer to the equilibrium solution. Equation 2.68 establishes such a path but so does an equation where Δl is any arbitrary fixed value that is small enough not to miss key path features. Accordingly, we simplify the constraint equation to

$$\Delta l^2 = \Delta q_{i+1}^T \Delta q_{i+1} \tag{2.69}$$

Unfortunately even this simple constraint equation has the effect of destroying the symmetry of our global stiffness matrix. This problem is overcome by returning to equation 2.67 and breaking δq_i into two parts.

$$\delta q_i = \delta q_{i1} + \delta\lambda_i \delta q_{i2} \tag{2.70}$$

where

$$\delta q_{i1} = -K_T^{-1} F(q_i, \lambda_i) \tag{2.71}$$

$$\delta q_{i2} = K_T^{-1} R \tag{2.72}$$

If we also break the updated displacement and load parameter increments into sums of the previous incremental values and the change between increments, we have

$$\begin{aligned} \Delta q_{i+1} &= \Delta q_i + \delta q_i \\ \Delta\lambda_{i+1} &= \Delta\lambda_i + \delta\lambda_i \end{aligned} \tag{2.73}$$

Now, by substituting these equations for Δq_{i+1} in the constraint equation 2.69 we get a quadratic equation in $\delta\lambda_i$ such that

$$a\delta\lambda_i^2 + b\delta\lambda_i + c = 0 \tag{2.74}$$

where

$$a = \delta q_{i2}^T \delta q_{i2}$$

$$\begin{aligned} b &= 2\delta q_{i2}^T(\Delta q_i + \delta q_i) \\ c &= (\Delta q_i + \delta q_i)^T(\Delta q_i + \delta q_i) - \Delta l^2 \end{aligned} \tag{2.75}$$

When the above system converges, i.e. δq_i becomes smaller than a user–defined convergence criteria, the total displacement q_m and the total load parameter λ_m for the mth load step are

$$q_m = q_{m-1} + \Delta q_m \qquad \lambda_m = \lambda_{m-1} + \Delta\lambda_m \tag{2.76}$$

and the search radius for an incremental load step is

$$\Delta l_m = \Delta l_{m-1}\frac{N_s}{N_{m-1}} \tag{2.77}$$

where N_s is a user-defined number of iterations and N_{m-1} is the number of iterations required to satisfy convergence for step $m-1$. Once we have Δl_m for each load step, the initial load increment or decrement parameter is found from

$$\Delta\lambda_1 = \pm\frac{\Delta l_m}{\sqrt{\delta q_{i2}^T \delta q_{i2}}} \tag{2.78}$$

where $\Delta\lambda_1$ is positive for a positive determinant of K_T and negative for a negative determinant. Finally, each load increment is begun with the linear solution

$$\Delta q_1 = \Delta\lambda_1 \delta q_{i2} \tag{2.79}$$

It is difficult by an algebraic derivation such as this to clearly represent how an actual numerical code for a Riks type solution might be executed. In this vein, an attempt at a clear, step–by–step algorithm is presented.

2.6 Step–by–Step Riks Algorithm

Refer to figure 2.7.

1. First increment, first iteration: compute only constant,K, terms of K_T.

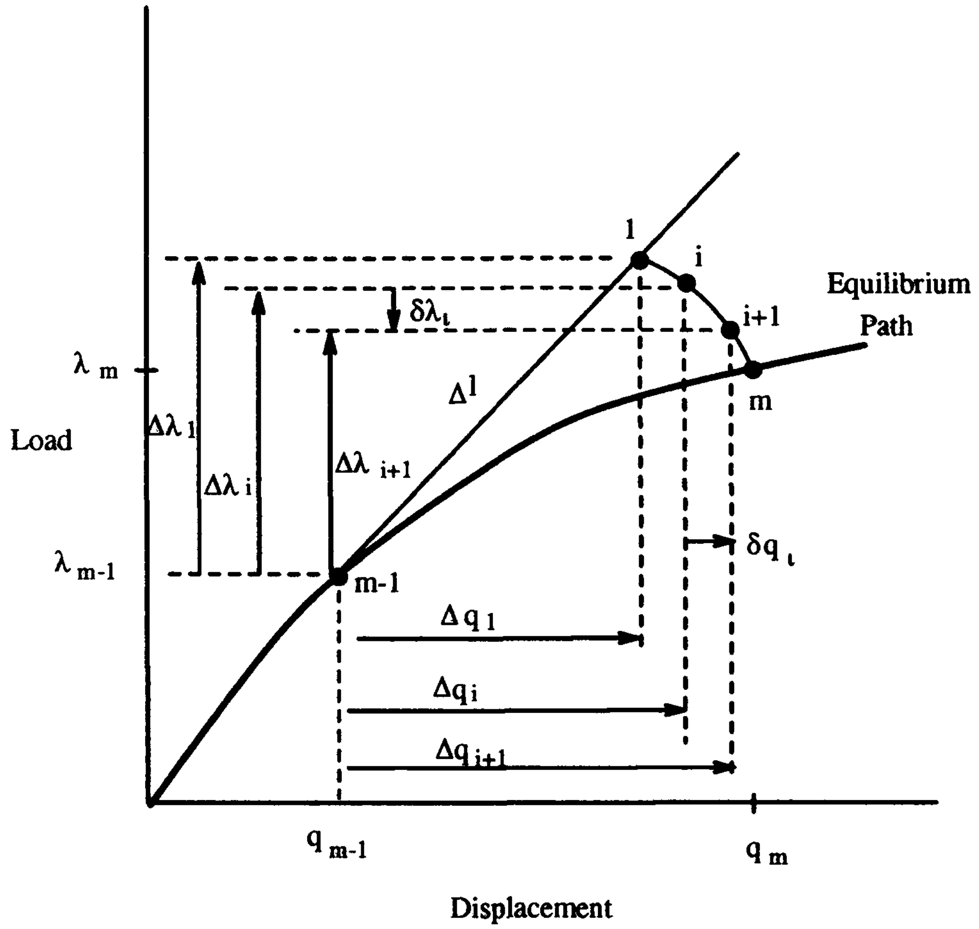

Figure 2.7. Riks Technique

2. First iteration, each load increment: compute $\delta q_{i2} = K_T^{-1}R$. For the first increment K_T has only constant terms; otherwise it contains the q_{m-1} terms.

3. Each iteration, each increment: compute $\Delta q_i = \Delta\lambda_i \delta q_{i2}$. On the first increment $\Delta\lambda_i = \lambda_0$ which is prescribed by the user. On the first iteration of each increment, $\Delta\lambda_i$ is $\Delta\lambda_1$.

4. Each iteration, each increment: compute in order,

$$\begin{aligned} \delta q_{i1} &= -K_T^{-1}F(q_i, \lambda_i) \\ \delta q_i &= \delta q_{i1} + \delta\lambda_i \delta q_{i2} \\ \Delta q_{i+1} &= \Delta q_i + \delta q_i \end{aligned} \tag{2.80}$$

5. Compute $\Delta l = \sqrt{\Delta q_{i+1}^T \Delta q_{i+1}}$.

6. Update K_T including $q_i = q_{m-1} + \Delta q_i$.

7. Solve quadratic equation $a\delta\lambda_i^2 + b\delta\lambda_i + c = 0$ (see equation 2.75 for a,b,c) for $\pm\delta\lambda_i$. If roots are complex, return to step 2 after arbitrarily adjusting Δl_m and, consequently, $\Delta\lambda_1$ to avoid complex roots.

8. Choose $\pm\delta\lambda_i$ based on which yields a positive θ where

$$\begin{aligned} \delta q_i &= \delta_{q1} \pm \delta\lambda_i \delta q_{i2} \\ \theta &= (\Delta q_i + \delta q_i)\Delta q_i \end{aligned} \tag{2.81}$$

and if both θ values are positive, $\delta\lambda_i = -c/b$.

9. Update the displacement and loading parameter

$$\begin{aligned} \Delta q_{i+1} &= \Delta q_i + \delta q_i \\ \Delta\lambda_{i+1} &= \Delta\lambda_i + \delta\lambda_i \end{aligned} \tag{2.82}$$

10. Check convergence criteria. If no convergence return to step 2. On convergence, update displacement and loading parameter for next increment.

$$\begin{aligned} q_m &= q_{m-1} + \Delta q_m \\ \lambda_m &= \lambda_{m-1} + \Delta\lambda_m \end{aligned} \tag{2.83}$$

11. Compute new search radius for next load increment

$$\Delta l_m = \Delta l_{m-1} \frac{N_s}{N_{m-1}} \tag{2.84}$$

where N_s is a user prescribed iteration estimate and N_{m-1} is the number of iterations required for convergence in the last load increment.

12. Compute loading parameter for the next increment, first iteration,

$$\Delta\lambda_1 = \pm \frac{\Delta l_m}{\sqrt{\delta q_{i2}^T \delta q_{i2}}} \tag{2.85}$$

13. Return to step 2 to start a new load increment.

III. Results and Discussion

3.1 Clamped–Clamped Shallow Arch

Our first problem is an isotropic shallow arch with both ends clamped and with dimensions as shown in figure 3.1. Here, shallow is defined as an arch where the rise–to–span ratio, δ/b, is less than 1/4. The arch is loaded by a point load at the top center. This problem has been explored by several previous investigators, however, Belytschko and Glaum (1) were the first to trace the post–buckling response and comparisons will be made with their work. Our first comparison, however, is between the present code and that developed by Palazotto and Dennis (from here on referred to with their term, SLR, for simplified large displacement/rotation). Results of the SLR run along with results from the present theory are shown in figure 3.2. The point load is plotted versus vertical displacement (down is positive) of the central point. As can be seen, comparison between the present work and SLR is excellent. These results confirm the validity of the shell–to–arch/beam assumptions made in the constitutive relations development. So, for a shallow, thin beam the present theory accurately predicts displacements and critical loads while being much more compact than the SLR code.

As mentioned, this symmetrically clamped isotropic arch was also investigated by Belytschko and Glaum (1) and their results are also shown in figure 3.2. The results of Belytschko and Glaum (B&G) are, obviously, initially much more flexible than the present work and the SLR solution. B&G used a higher–order corotational stretch theory to derive their curved beam elements. Three major differences exist between the present curved beam theory and that of B&G. First, while the present theory uses total Lagrangian kinematics, B&G use updated Lagrangian kinematics where the element's corotational coordinate system rotates through each solution increment to track rigid body motion. Second, the present work retains all nonlinear strain terms from the Green's strain tensor. B&G, on the other hand, exclude some nonlinear terms. Compare the in–plane strain from the present work,

$$\epsilon_2^0 = v_{,2} - wc + 1/2\left(v_{,2}^2 + w_{,2}^2 + v^2c^2 + w^2c^2\right) + vw_{,2}c - v_{,2}wc \qquad (3.1)$$

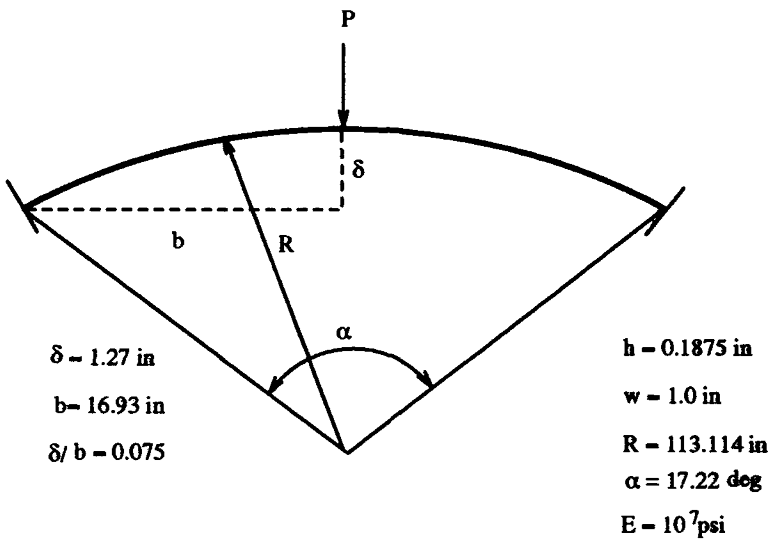

Figure 3.1. Clamped–Clamped Shallow Arch

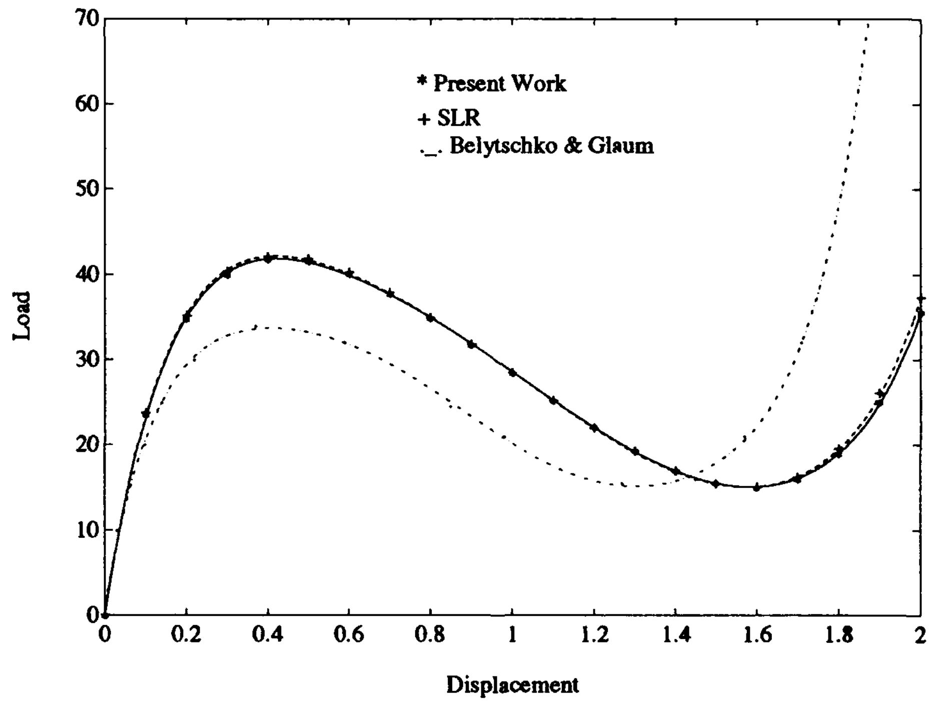

Figure 3.2. Comparison of present theory, SLR and B&G for clamped–clamped shallow arch

with B&G's

$$\hat{\epsilon}_x^{mid} = \frac{\partial \hat{u}_x^{def}}{\hat{x}} + \frac{1}{2}\left(\frac{\partial \hat{u}_y^{def}}{\partial \hat{x}}\right)^2 + \hat{\omega}_0 \frac{\partial \hat{u}_y^{def}}{\partial \hat{x}} \tag{3.2}$$

where the $\hat{}$ indicates with respect to the corotational coordinate system. The term $\hat{u}_x^{def}$ is deformation along the beam and corresponds to our v. The term $\frac{\partial \hat{u}_y^{def}}{\partial \hat{x}}$ is rate of vertical deformation along the beam length and corresponds to our $w_{,2}$. The term $\hat{\omega}_0$ accounts for separation of the beam centerline from the element corotational axis and has no correspondence to any of the terms in equation 3.1. Finally, B&G use Euler–Bernoulli beam theory to compute out of plane strains so, unlike the present work, transverse shear is not considered.

So, the present theory gives initially stiffer results than the corotational stretch theory because more nonlinear in–plane strain terms are retained. As displacements become large, however, the present theory produces more flexible results as the ability of the nonlinear strain terms to capture extensibility of the midplane becomes dominant.

Note that, for this problem, twenty increments (seen here as points on the graph) were necessary to plot a smooth equilibrium curve. The curves through the data points are seventh order best–fit polynomials computed and plotted in MATLAB. This technique of graphing the equilibrium curve is useful for the gentle slopes with smooth transitions through limit points. However, as will be seen in later problems, more complex equilibrium paths defy polynomial best–fits. For this reason and since curves with sharper transitions require small load steps many more increments were used on deep arch problems. This, of course, increases processing time but no effort was made to trade accuracy for shorter CPU times. It is popular in the literature to report short processing times; however, the nature of this work is not to get the quickest answer, only the best answer.

This clamped–clamped shallow arch problem was also used to consider convergence of the finite element model. Since the finite element uses Hermitian shape functions to interpolate vertical displacement w and slope $w_{,2}$ within each element, and that slope is continuous at the nodes, our elements are said to have C^1 continuity (3:99–100) for w. For the other degrees of freedom, we have C^0 continuity. Cook states three requirements for convergence of a finite element model for a generic field variable ϕ (3:126–127).

1. Within each element, the assumed field for ϕ must contain a complete polynomial of degree m, where m is the highest order of ϕ used in deriving the governing differential equation of the problem.
2. Across boundaries between elements, there must be continuity of ϕ and its derivatives through order $m - 1$.
3. Finally, if the elements are used in a mesh with boundary conditions such that the mth derivative of ϕ displays a constant value, then, as the mesh is refined, each element must come to display that constant value.

In our case, m is two and ϕ is w and the first two conditions are met since the Hermitian shape functions are cubic polynomials. For the third requirement, no such test was performed. Dennis and Palazotto point out that the 2–D element used in their shell theory passes the patch test, which assures convergence (13:82–83). No simplifications have been made to the interpolation functions in adapting this theory from SLR (excepting, of course 2–D to 1–D), so finite element solutions are expected to converge to the correct answer as more elements are added. In fact, for our clamped–clamped arch, the model does converge to a solution as the mesh is refined from two to nine elements as shown in figure 3.3.

3.2 Cantilevered Composite Beam

The next problem considered is one considered by Minguet and Dugundji (12) in their investigation by experiment and analysis of large deflections for composite beams resembling helicopter rotor blades. The beam analyzed is shown in figure 3.4. Minguet and Dugundji (M&D) formulated an updated Lagrangian displacement scheme based on Euler angles which track the rigid body motion of the element and are the arguments for a coordinate transformation from local to global coordinates. This displacement representation is exact with respect to the rigid body kinematics of an element since no approximations of trigonometric functions are made. M&D base their finite–difference solution on infinitesimal beam cross–sections and, as a result, all force quantities evaluated at a section are independent of the thickness dimension. Force equilibrium is enforced at the nodes as opposed to displacements in the present analysis. So, transverse shear is constant through the

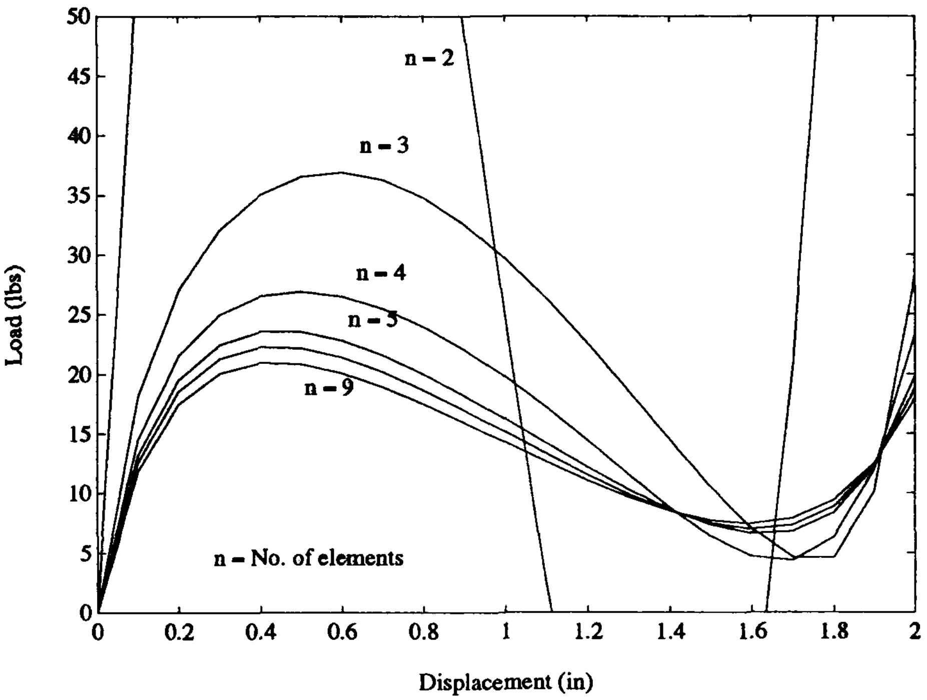

Figure 3.3. Convergence Test for Clamped–Clamped Arch

cross section, that is, the theory includes first–order transverse shear. Extensibility is permitted in M&D's analysis and the axial strain is coupled with the shear strains, twist rate and bending curvatures through stress–strain relations. While Minguet and Dugundji's analysis is considerably different from the present analysis in many respects and comparisons are difficult, the experimental data that is presented is very useful to test the ability of the present theory to predict laminated beam behavior. The experimental setup consisted of a AS4-3501/6 graphite epoxy $[0/90]_{3s}$ flat cantilever beam with dimensions as shown in figure 3.4. Composite material properties were: E_1 = 142GPa, E_2 = 9.8GPa, $G_{12} = G_{13}$ = 6GPa, G_{23} = 4.8GPa, ν_{12} = 0.3, h = .124mm and $\rho = 1580 kg/m^3$. A weight was applied at the beam tip, however, displacements were observed 50 mm from the tip.

In the present analysis displacement control was used with 33 elements, a convergence tolerance (TOL) of 0.002, and 5 mm vertical displacement increments at the tip. Many elements were required, in this case, because of the large displacements asked for in this problem.

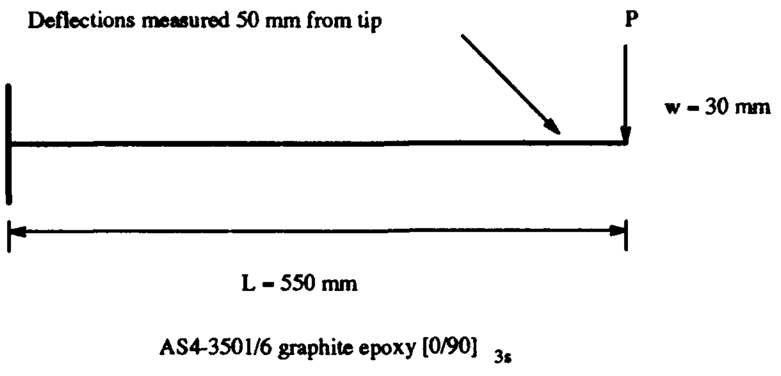
Deflections measured 50 mm from tip
P
w - 30 mm
L - 550 mm
AS4-3501/6 graphite epoxy [0/90] 3s

Figure 3.4. Cantilevered Composite Beam

w	$w_{,2}$	$w_{,2}$ in degrees	tan $w_{,2}$	% error
0.11	0.353	20.2 deg	0.368	4
0.15	0.482	27.6 deg	0.523	8
0.20	0.660	37.8 deg	0.776	16

Table 3.1. Angular Estimation Error for Cantilever Beam Problem

The shape of the beam throughout the deflection is shown in figure 3.5. A comparison of results for the present theory versus Minguet and Dugundji's is presented in figure 3.6. The experimental and analytical results of M&D were indistinguishable in their article; so, only one line is shown here. As can be seen, the present theory virtually coincides with M&D's results through vertical displacement,w, of about 0.1 meters. Part of the difference here is directly attributable to the kinematics of the present theory. The present work uses radians for all angular measure and at appreciable angles of rotation, such as those in this problem, the difference between the angle in radians and the tangent function of the angle becomes significant. M&D follow the real kinematics more accurately since they use an exact transformation matrix from elemental to global coordinates. Table 3.2 shows how the tangent of the angle differs from radian measure for some relevant values. As the beam deflects to 0.15 meters, the error is over 20%; angular error, however, is only about 8% at this point. The remainder of the error is due to higher–order terms in the in–plane strain of the present solution. That is, in a nonlinear system such as this errors can easily become magnified as these terms become squared or multiplied with other terms. This is especially true in a theory such as this which has strain–displacement relations as its foundation. Now

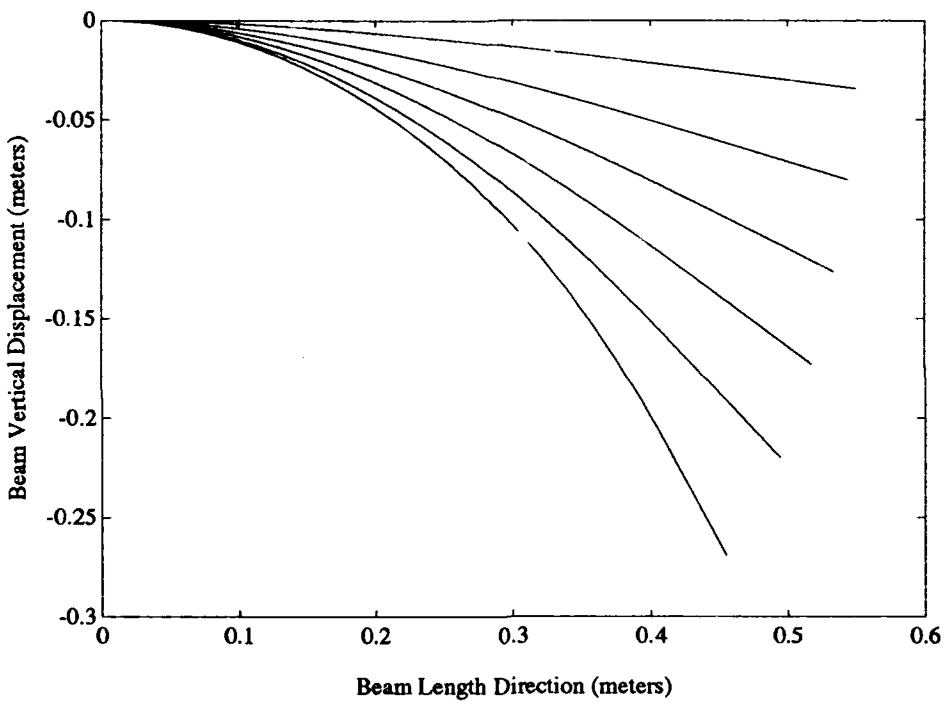

Figure 3.5. Shape of Deflected Cantilevered Composite Beam Through Six Displacement Increments

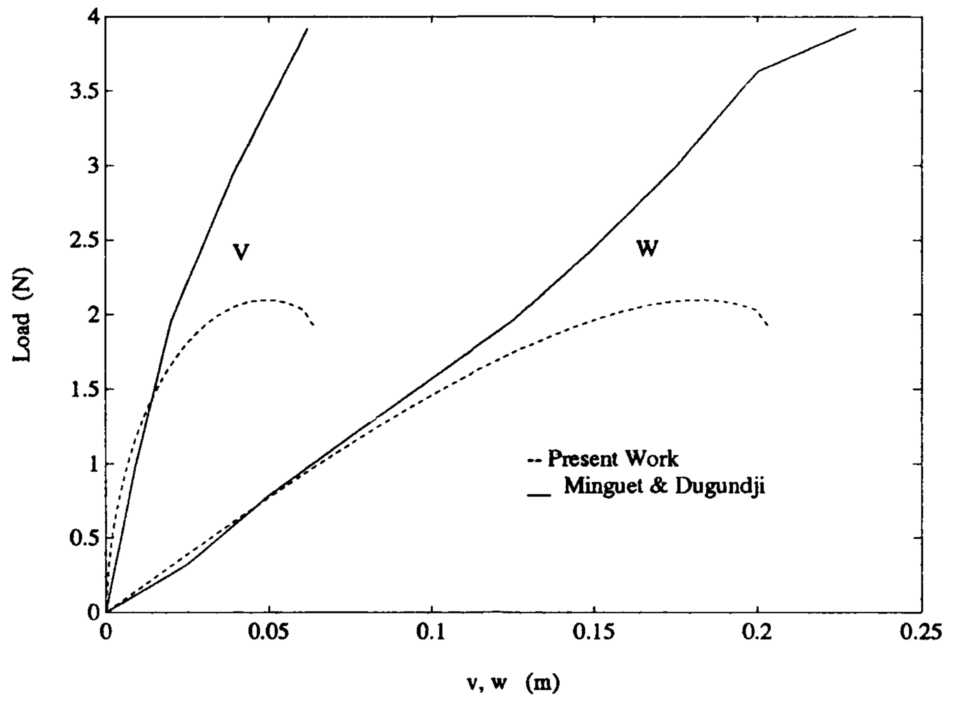

Figure 3.6. Cantilevered Composite Beam–Results

this theory has been shown to accurately represent large displacements of cylindrical shells; why the difficulty in this problem? In most previous analyses, this theory has been used to represent structures that, if not symmetrical, have mass on either side of the applied load and the applied load remains transverse to the shell midplane. In these cases the stretching is restrained and that strain energy is transferred to bending and transverse shear modes. In this case, however, the free end of the cantilever is not restrained from stretching and the load approaches being parallel to the beam at large deflections. Also consider the fact that, since the rotational terms $w_{,2}$ and ψ_2 are underestimated by using radians, the finite element system of equations must shift strain energy to the remaining degrees of freedom, v and w, to achieve equilibrium. Hence, underestimation of the rotations leads to more flexible results for v and w. This is clearly represented in the results where v and w depart the M&D experimental/analytical curves at roughly the same load level.

So, in cases where mass is not present on both sides of the load and for rotations above about 20 degrees, the present theory can accumulate large displacement errors due to higher–order representation of these terms in the strain–displacement equations. This situation is somewhat ironic since these higher–order terms are included to more exactly represent these strains. And the theory does well in cases where the in–plane strains are due primarily to coupling with the bending and shear modes; that is, it is an excellent arch theory but it is not well suited to problems such as this one where stretching approaches bending and shear as a primary displacement mode. It was known when this theory was developed that the overall strain field was incompatible (13:36–37)in that we have nonlinear in–plane strains, linear transverse shear strains, and zero transverse normal strain. So, Palazotto and Dennis beefed up the midplane kinematics to be very sensitive to coupling with other modes, not to be the primary displacement mode. This problem does, however, demonstrate clearly that this theory accurately represents the stiffness of symmetrically stacked laminates.

3.3 Cantilever with Tip Moment

The next problem considered has been previously investigated by Epstein and Murray (9). The problem consists of an isotropic cantilevered straight beam subjected to a tip moment as shown in figure 3.7. Displacement control was used to impose rotations at the beam tip. After tip rotations of 53 degrees at a resulting moment of 14.5 lb-in, the solution stopped producing reasonable results. Moments were computed based on the analytical results for w and $w_{,2}$ at the beam tip and utilizing the Hermitian shape functions of equation 2.55 to calculate $w_{,22}$. Then the moment is

$$M = EIw_{,22} \tag{3.3}$$

where

$$w_{,22} = \frac{1}{a^2}\left[H_{11,\eta\eta}w^{(1)} + H_{12,\eta\eta}w_{,2}^{(1)} + H_{21,\eta\eta}w^{(2)} + H_{22,\eta\eta}w_{,2}^{(2)}\right] \tag{3.4}$$

Figure 3.8 presents comparisons between the current effort and that of Epstein and Murray. Results of the present work diverge from those of Epstein and Murray at a tip rotation of about 20 degrees, or, as is seen more easily in the figure, at a moment of about

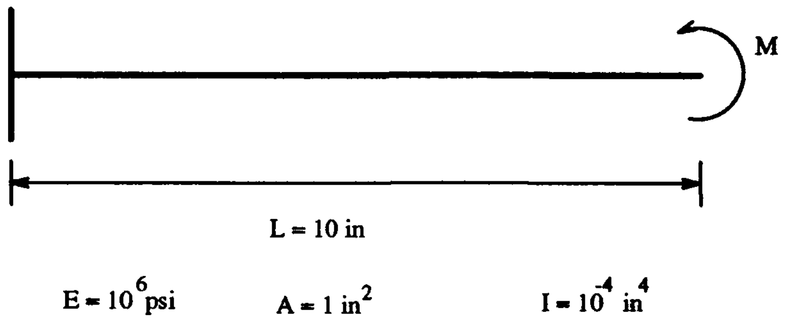

Figure 3.7. Cantilever with tip moment

5 lb-in. This is in agreement with the previous problem where the current results diverged from those of Minguet and Dugundji at 20 degrees.

Like the present research, Epstein and Murray (E&M) use a total Lagrangian approach. However, E&M retain all trigonometric terms in their vectorially based kinematics and, as a result, can tolerate much larger rotations than the present theory. For instance, E&M's equilibrium equations for an element may be represented as

$$\begin{aligned} T\cos\phi + S\sin\phi &= (2e+1)\mathcal{N} \\ -T\sin\phi + S\cos\phi &= -\frac{[(2e+1)\mathcal{M}]}{\sqrt{2e+1}} \end{aligned} \tag{3.5}$$

where S and T are x and z components of the external end force, ϕ is the rotation angle of the tangent to the beam axis, e is Green's strain along the axis, $\mathcal{N}$ is the internal force resultant and $\mathcal{M}$ is the internal moment resultant.

Figure 3.9 shows the shape of the deflected beam through solution increments, but presents only part of E&M's results. In fact, E&M were able to bend the beam into a complete circle (9:8).

Epstein and Murray do not include transverse shear in their model. This is not important in this problem since, in order to duplicate the cross-sectional properties specified in the problem a very thin beam (0.03464 inches) was necessary.

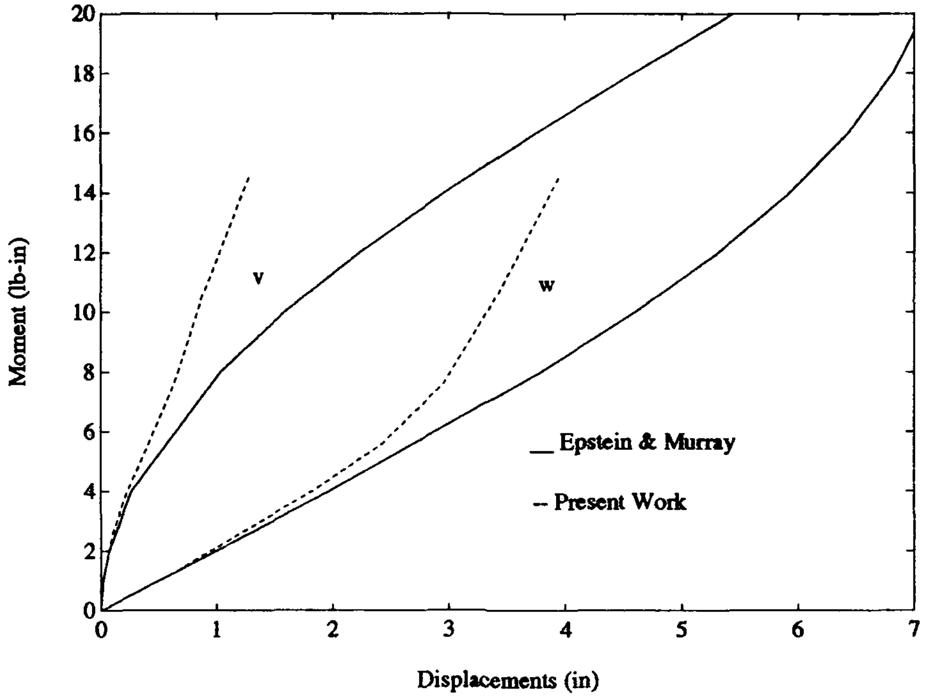

Figure 3.8. Cantilever with tip moment–Results

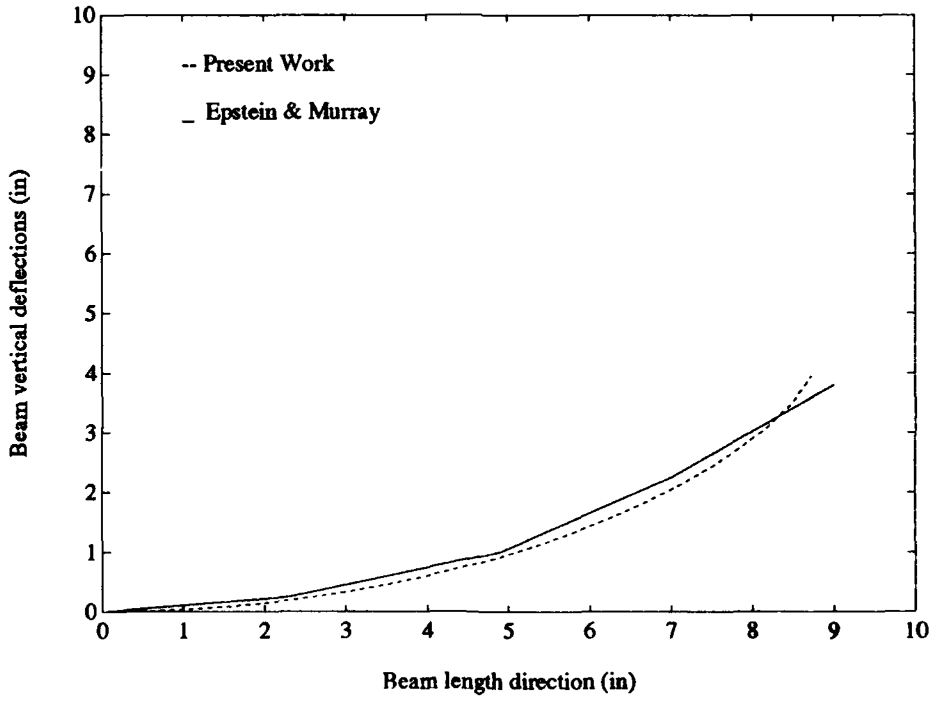

Figure 3.9. Cantilever with tip moment–deflected shapes

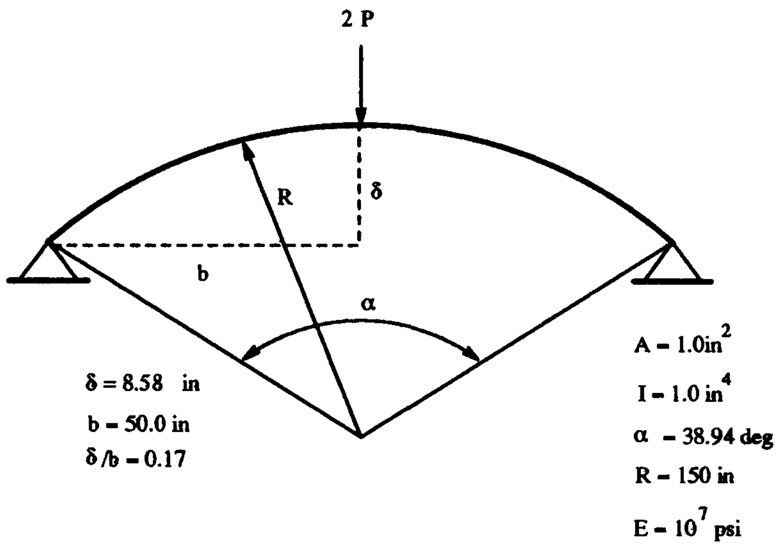

Figure 3.10. Hinged–Hinged Shallow Arch

3.4 Hinged–Hinged Shallow Arch

The next problem considered is a hinged–hinged arch previously considered by Sabir and Lock (21). The geometry and material characteristics of the isotropic arch are shown in figure 3.10. This is an interesting problem in two respects. First, this arch displays an equilibrium path with both snap–through and snap–back limit points. So, displacement control is of limited use; Riks technique is required to trace the entire equilibrium path. Second, in order to get the area and area moment of inertia specified by Sabir and Lock and shown in figure 3.10 a thickness of 3.46 inches was required. This results in a thickness–to–radius ratio of 1/43. This is a relatively thick beam and transverse shear could become important.

Riks technique was used with the present analysis with the following parameters, $\lambda_0 = 0.1$, $N_s = 2.5$, $M_s = 2.0$ and $TOL = 0.2$

As the results indicate, the present work with Riks technique follows the equilibrium path through four horizontal limit points and two vertical limit points. The labeled points in figure 3.11 correspond to plots of the arch deflected shape depicted in figure 3.12. Point A is a local snap–through point and this action is apparent in figure 3.12 where the center portion of the arch has reversed curvature. As the crown of the arch deflects downward to point B, load shedding takes place. In fact, as the equilibrium path descends into negative

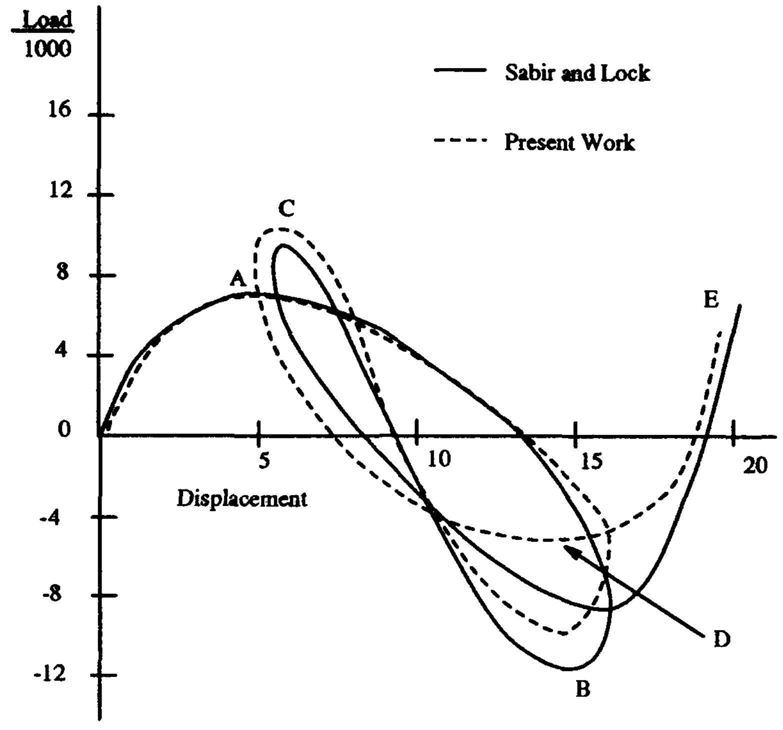

Figure 3.11. Hinged–Hinged Shallow Arch Equilibrium Path

load the load mechanism must pull up on the arch to achieve equilibrium. This sequence from *A* to *B* is dynamic in the sense that, in an experimental trial, the snapping would occur instantaneously. In the curve portion from *B* to *C*, the arch crown deflects upwards as the ends begin reversing curvature and the arch supports increased loads. At point *C* the ends have completely reversed curvature; however, the crown is again in a positive curvature position. Between points *C* and *D* the structure again sheds load and the crown again reverses curvature. Finally, at point *D*, all local and global curvatures are reversed and the arch is stable since it is able to accept increasing loads while displacing in a positive sense.

Sabir and Lock do not represent extensibility to the degree of the present work. Longitudinal strains are represented as

$$\epsilon_y = v_{,y} + \frac{w}{R} + \frac{1}{2}w_{,y}^2 - z(w_{,yy} - \frac{v_{,y}}{R}) \tag{3.6}$$

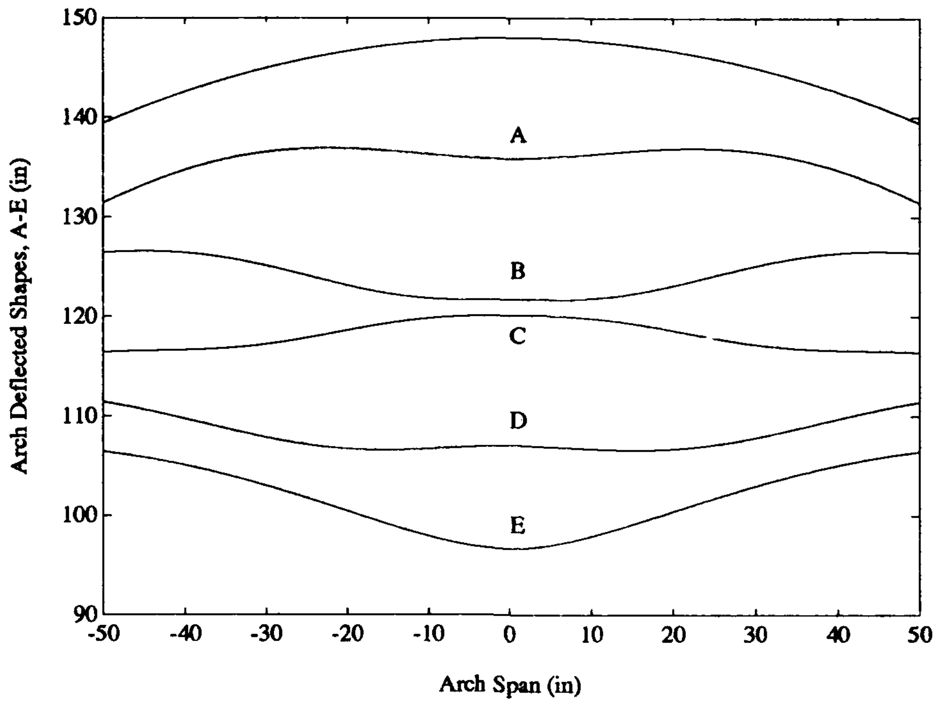

Figure 3.12. Deflected Shapes of Hinged–Hinged Shallow Arch

This relationship contains more higher order terms than most theories; however, our Green's strain representation includes many more terms which are a function of the thickness coordinate, z. This is especially important in this problem since the arch is fairly thick. Indeed the higher order rotational terms so prominent in our ϵ_2 representation do become significant in this problem. At point D in figure 3.11 the slope of the elastic curve, $w_{,2}$ is -0.159 radians while the bending rotation, ψ_2 is 0.193 radians. The algebraic sum, 0.034 radians or 1.94 degrees, is the rotation at the midplane due to transverse shear strain, β. These higher–order rotational terms appear in our midplane strains which, in this case, are restrained by the symmetric boundary conditions. The result is a stiffer structure which is verified by the results since the plot of the present results lies above that of Sabir and Lock at large deflections. It is also important to point out, in general for nonlinear problems, the history of the equilibrium path is a factor. Once two solutions diverge for whatever reason, since subsequent data points depend on strain energies present at the

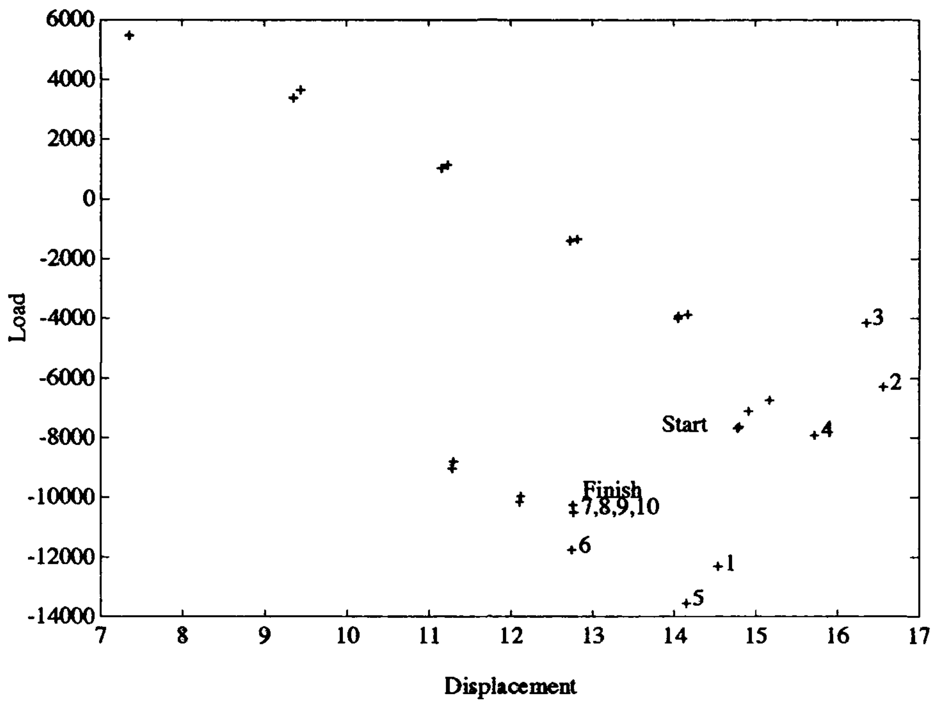

Figure 3.13. Iterations of the Riks Algorithm

previous solution step, there is no reason to expect the solutions to rejoin the same path again.

This problem was also used to present a demonstration of the numerical convergence characteristics of the Riks algorithm. Figure 3.13 shows a close–up of the same curve as in figure 3.12 except individual data points at each iteration within seven increments are shown. The increment under inspection begins with the previous converged value, the double cross just right of "Start". The first iteration produces point 1 and iteration continues on through 10 iterations until convergence is achieved at the clustered points labeled "Finish". It's important to realize that no neat "search radius" or arc length is apparent here because we are seeing a projection onto one degree of freedom of an $n+1$ dimensional surface, where n is the number of degrees of freedom. The search radius, Δl, so carefully described in Chapter 2 does exist in $n+1$ dimensional space; it's unrecognizable on this plane though.

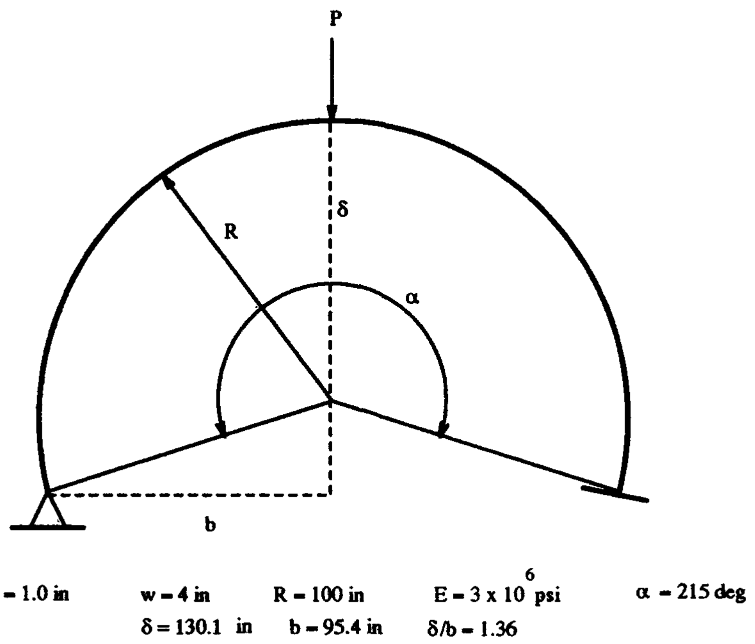

Figure 3.14. Hinged–Clamped Very Deep Arch

3.5 Hinged–Clamped Very Deep Arch

Very deep arches are those where the subtended angle, α, exceeds 180 degrees. This geometry is particularly challenging for large displacement/large rotation theories since large rotations are required to reach interesting areas (i.e. limit points) of the load versus displacement curves. The hinged–clamped isotropic deep arch shown in figure 3.14 has been investigated by Brockman (2) and by DaDeppo and Schmidt (7). The unsymmetric boundary conditions in this problem represent an additional challenge for our theory since very large rotations are possible near the hinged end of the arch. Brockman uses a 3–D finite element code and updated Lagrangian kinematics to trace the equilibrium path. Transverse shear is not included in Brockman's solution. DaDeppo and Schmidt, on the other hand, use a total Lagrangian formulation which is exact since no angle approximations are used. In this case, DaDeppo and Schmidt assumed an inextensible midplane and did not consider transverse shear. For this problem, the Riks approach was used after displacement control failed at displacements well below those observed by Brockman and DaDeppo & Schmidt. Eighty equal length elements were used with a tight convergence criteria of 0.001 %. The estimated number of iterations, N_s, was set to 2.2 in an effort

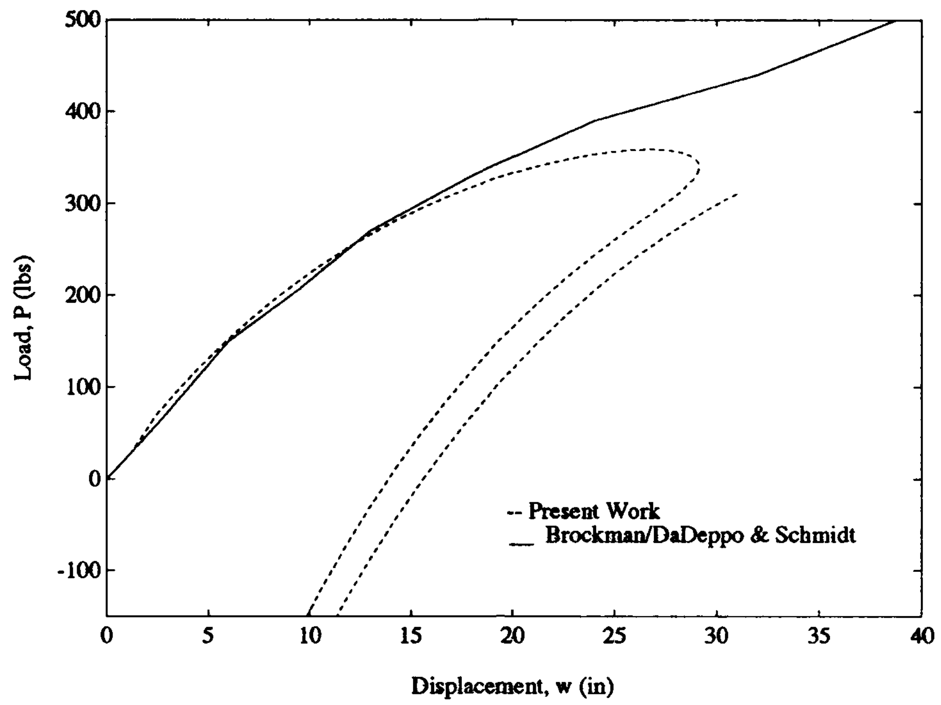

Figure 3.15. Hinged–Clamped Very Deep Arch–Vertical Displacement versus Load

to closely follow the equilibrium path. However, as the results indicate in figure 3.15, the present theory with the Riks algorithm ran into difficulty well below displacements achieved by the previous investigators. Our theory diverged from Brockman/DaDeppo & Schmidt at a crown vertical displacement of 17 inches. At this point, rotations at the crown were only 9.7 degrees; however, rotations at nodes on the hinged side of the arch were as high as 23 degrees. This corresponds with rotation limits of the present theory seen in previous problems. At a crown vertical displacement of 27 inches, the present theory encountered a limit point which was not observed by the previous investigators. At this alleged limit point, rotations at the arch crown were 16 degrees and a maximum nodal rotation of 34.5 degrees was observed on the hinged side of the arch. The original shape of the arch and the deflected shape at the limit point encountered are shown in figure 3.16.

The limit point shown in figure 3.15 resembles bifurcation points encountered by Tsai and Palazotto in their previous work with shells using Riks technique (23). The dip in the plot of the present results is not fully represented in the figure for clarity. The curve

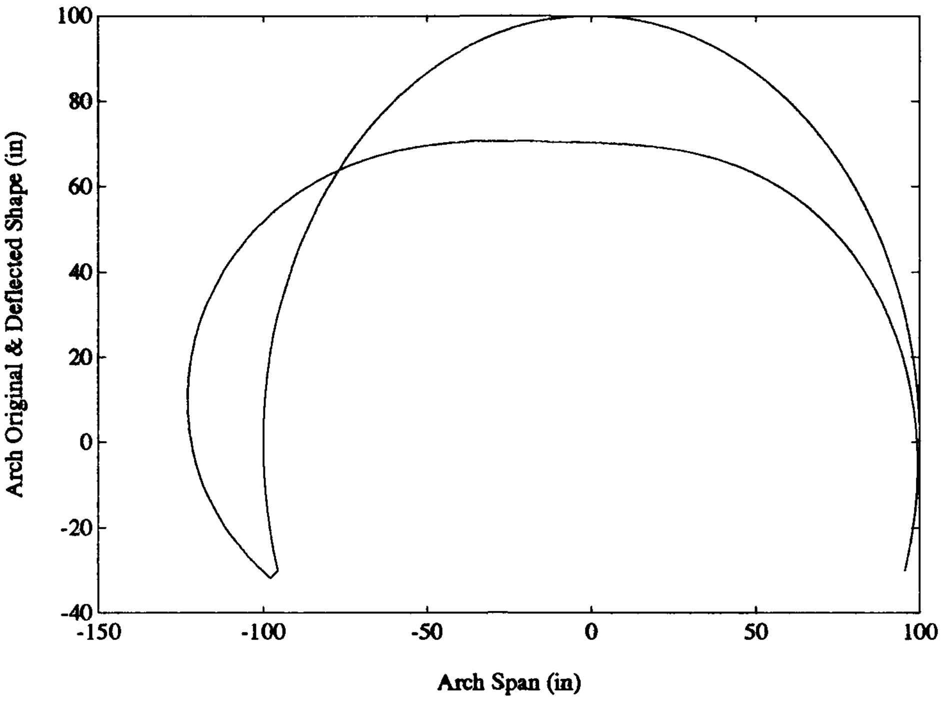

Figure 3.16. Hinged–Clamped Deep Arch–Original and Deflected Shapes

actually extends to a minimum of-900 pounds load before reversing and terminating at the final point shown. Full representation for the observations of Brockman and DaDeppo & Schmidt are also not shown. Brockman was able to follow the equilibrium path, and encountered no limit points, through 113.7 inches (2:6.10.1). DaDeppo & Schmidt were able to trace the equilibrium path to a vertical displacement slightly above that of Brockman where they encountered a limit point.

This deep arch problem clearly reveals the rotational limitations of the current theory. The current solution agreed closely with other solutions until rotations in the arch began to exceed 23 degrees. At this point vertical deflections were 17 times the arch thickness.

3.6 Hinged-Hinged Deep Arch

The next problem considered is an isotropic deep arch described in figure 3.17. This arch has been investigated by Huddleston (10) and by Dennis (8). Dennis utilized the SLR

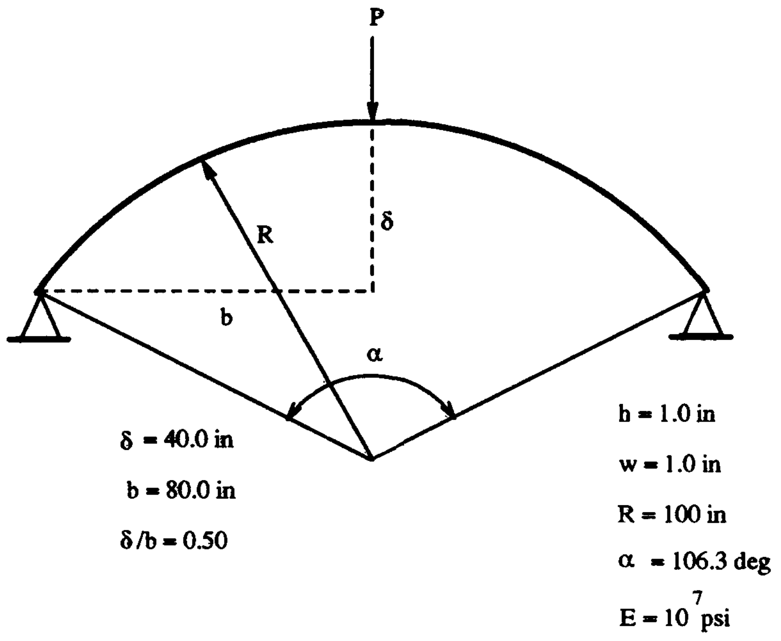

Figure 3.17. Hinged–Hinged Deep Arch

theory of Dennis & Palazotto and also looked at simplifications implied by making Donnell type assumptions.

The present theory was applied with both displacement control and Riks technique. Thirty elements were used to model half of the symmetric arch. Boundary conditions at the crown consisted of fixing ψ_2, v, and $w_{,2}$ to zero and allowing w to be controlled by specified displacements in the displacement control technique and free for the Riks technique. For both cases a convergence tolerance of 0.02% was used and, for the Riks technique, an N_s of 2.2 was used.

As is seen from the results in figure 3.18 the present theory agreed very well with the SLR theory throughout the extent of the solution. The present solution is slightly less flexible than SLR at the upper tail of the curve; this is likely due to the inclusion in the present theory of thirteen additional higher–order terms for the in–plane strain that were ignored in SLR. The plot of the present theory is for displacement control; however, the Riks algorithm produced an identical equilibrium path.

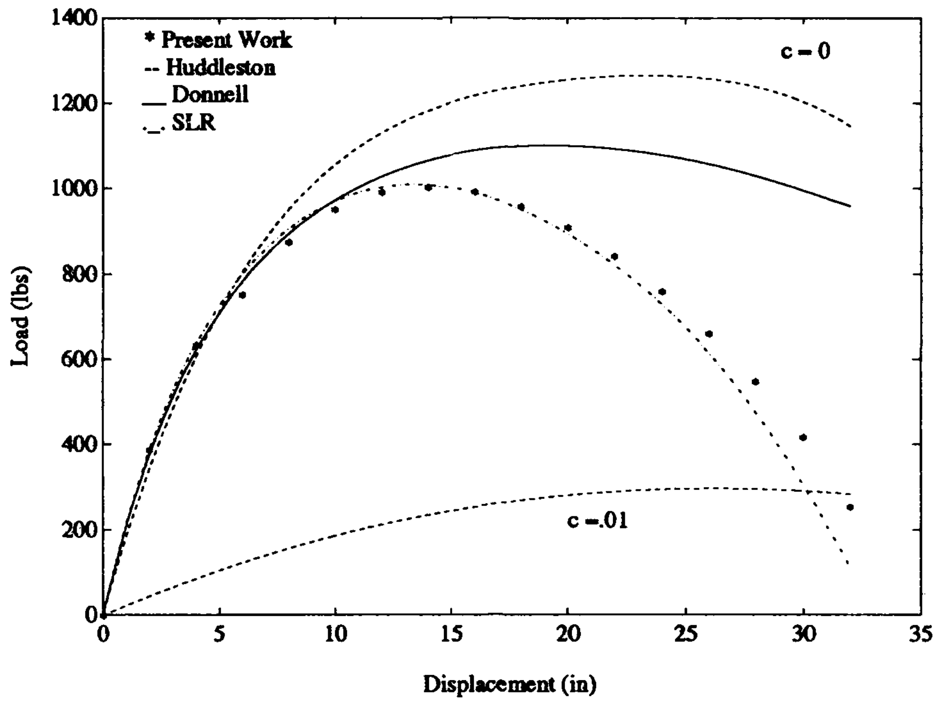

Figure 3.18. Hinged–Hinged Deep Arch Results

Dennis and Palazotto explain differences between the SLR results and the Huddleston results and Donnell assumption results in their text (13:205:206). Those explanations also apply to the present theory and they are reiterated here.

The Huddleston solutions presented include closed–form solutions for extensible and inextensible midplanes. Huddleston defines extensibility by a compressibility parameter (10:765), c, which is a ratio of bending stiffness to axial stiffness for an arch.

$$c = \frac{I}{A(2b)^2} \tag{3.7}$$

where I is the cross–sectional area moment of inertia, A is the cross–sectional area and, here, b is the horizontal distance from the center of curvature to either of the supports. The $c = 0$ solution presented is Huddleston's inextensible result and, not surprisingly, it is the stiffest of all solutions considered. The $c = 0.01$ solution represents the same arch geometry but with the cross–section varied to decrease axial stiffness and it is the most flexible solution presented. The cross–section and geometry of the arch considered here result in $c = 3.255x10^{-6}$ which is close to zero so it is reasonable that the SLR/present results closely follow the inextensible $c = 0$ curve for smaller displacements. Beyond this point the SLR/present results pass a snap-through limit point and are much more flexible than the inextensible results. Flexibility compared to the Huddleston inextensible solution even for such a small c may be explained by the fact that Huddleston does not include higher-order rotational terms in his midplane extensibility.

The assumptions used by Dennis to yield the Donnell solution result in elimination of many higher-order terms in the midplane strain relations. So, the SLR/present solutions are logically more flexible since, as deflections increase, the higher-order rotational terms become important.

At the snap-through point, the present solution resulted in maximum nodal rotations of 23.9 degrees. This value exceeds that seen to cause significant error in previous problems considered in this work. So, some of the flexibility displayed in the SLR/present solutions may be due to angular estimation error. The present solution is numerically viable through displacements of 30 inches where the maximum nodal rotation was 47 degrees. The shape of the arch in the initial configuration, at the snap-through point and at the maximum deflection attained is shown figure 3.19. Beyond this point, displacement control could not converge to a solution and Riks technique produced a spurious bifurcation point similar to that encountered in the very deep arch problem above.

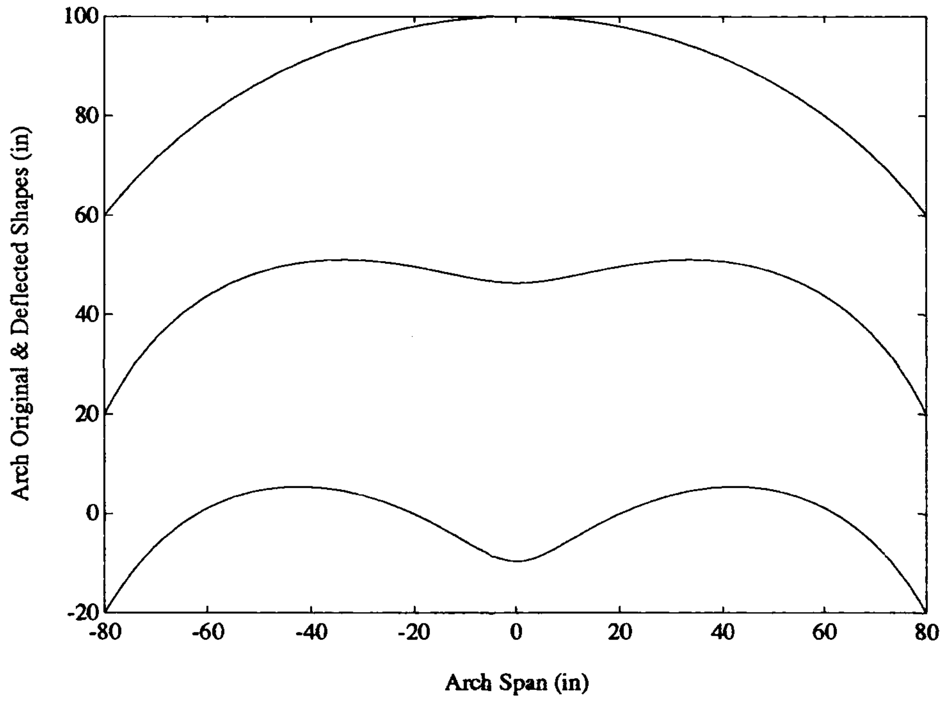

Figure 3.19. Hinged-Hinged Deep Arch Initial and Deflected Shapes

IV. Conclusions

This effort led to successful reduction of the two–dimensional geometrically nonlinear cylindrical shell theory of Dennis and Palazotto to a one–dimensional theory. The theory applies to large displacements and moderate rotations of isotropic or symmetric laminate arches or straight beams. The theory incorporates all Green's displacement strain terms for the midplane strains but only linear displacement terms for the transverse shear strains. The transverse shear strains are represented as a parabolic distribution through–the–thickness. This representation meets the boundary conditions of zero strain on the top and bottom surfaces and, hence, avoids shear locking and obviates the need for reduced integration or shear correction factors. The theory resulted in nonlinear differential equations which were converted to nonlinear algebraic equations via incorporation into a finite element scheme. A new finite element was developed which works equally well for arches and straight beams.

A variety of problems were run to explore the applicability and limitations of the theory. All problems investigated, save the laminated cantilever with tip moment, were analytical solutions. No experimental data exists for the geometrically nonlinear arch problems which snap–through and snap–back since the static equilibrium conditions are impossible to generate. So, comparisons were between various analytical solutions for beams and arches.

A convergence test was performed for a shallow arch problem. As the number of finite elements was increased, the model converged to the solution from above as is expected of a proper finite element scheme.

Displacement control and Riks method were found to be accurate and versatile in tracing nonlinear equilibrium paths. Displacement control easily passed snap–through points and Riks method was able to pass snap–through as well as snap–back points. A plot was constructed of iterations within one Riks increment for a ten iteration increment. Normally, though, only two iterations were required depending on the tolerance specified. Also, for the Riks technique, this author found it useful to vary the estimated number of iterations, N_s, as a real number to closely control increment sizes throughout a solution. Previous investigators had only used N_s as an integer.

A cantilever symmetric laminate beam with a tip load was investigated and, through rotations of twenty degrees, the present theory accurately predicted experimental results. An isotropic cantilever was subjected to a tip moment and, again, the current theory agreed with other analytical data through rotations of twenty degrees. Beyond this limit the present solution was more flexible than the comparison solutions and expected actual beam response.

Several arch problems, including shallow, deep and very deep arches were investigated and the current theory closely agreed with other analytical results for all arches through rotations of 23 degrees. Beyond this limit the present solution was more flexible than the comparison solutions and the comparison solutions are likely more accurate. For shallow arches, however, the present theory is viable for all rotations.

Comparisons with Huddleston's inextensible and extensible solutions, in particular, resulted in close agreement between the present solution and the *inextensible* at small displacements. As displacements became large, though, the present solution eventually intersected the extensible solution.

In conclusion, a successful one–dimensional reduction of the previous shell theory was completed. The rotational limits of the SLR shell theory were explored without the interference of 2–D coupling. While exploring these limitations, various other nonlinear beam/arch theories were found to be more accurate at representing large rotations. However, as has been mentioned, creating an exact large rotation beam/arch theory was not a goal of this research. A verified FORTRAN code was generated and utilized to investigate the limits of the theory. The limiting factor in the present theory is the approximation of rotations by directly applying radian measure angles in the kinematic relations rather than using trigonometric functions of those angles.

Appendix A. $\bar{Q}_{ij}, H, L, S, \hat{K}, \hat{N}_1, \hat{N}_2$

Transformation of Q_{ij}'s to $\bar{Q}_{ij}$'s of equation 2.3 for ply orientation angle θ from the global 1 axis (22):

$$
\begin{aligned}
\bar{Q}_{11} &= Q_{11}\cos^4\theta + 2(Q_{12}+2Q_{66})\sin^2\theta\cos^2\theta + Q_{22}\sin^4\theta \\
\bar{Q}_{12} &= (Q_{11}+Q_{22}-4Q_{66})\sin^2\theta\cos^2\theta + Q_{12}(\sin^4\theta+\cos^4\theta) \\
\bar{Q}_{22} &= Q_{11}\sin^4\theta + 2(Q_{12}+2Q_{66})\sin^2\theta\cos^2\theta + Q_{22}\cos^4\theta \\
\bar{Q}_{16} &= (Q_{11}-Q_{12}-2Q_{66})\sin\theta\cos^3\theta + (Q_{12}-Q_{22}+2Q_{66})\sin^3\theta\cos\theta \\
\bar{Q}_{26} &= (Q_{11}-Q_{12}-2Q_{66})\sin^3\theta\cos\theta + (Q_{12}-Q_{22}+2Q_{66})\sin\theta\cos^3\theta \\
\bar{Q}_{66} &= (Q_{11}+Q_{22}-2Q_{12}-2Q_{66})\sin^2\theta\cos^2\theta + Q_{66}(\sin^4\theta\cos^4\theta) \\
\bar{Q}_{44} &= Q_{44}\cos^2\theta + Q_{55}\sin^2\theta \\
\bar{Q}_{45} &= (Q_{44}-Q_{55})\cos\theta\sin\theta \\
\bar{Q}_{55} &= Q_{55}\cos^2\theta + Q_{44}\sin^2\theta
\end{aligned}
\tag{A.1}
$$

The L and S vectors and H matrices from equations 2.49 and 2.51 follow. In these terms, $c = 1/R$ and $k = -\frac{4}{3h^2}$.

$$
\begin{aligned}
L_0^T &= \{0\ 1\ -c\ 0\ 0\ 0\ 0\} \\
L_1^T &= \{0\ 0\ -c^2\ 0\ 0\ 0\ 1\} \\
L_2^T &= \{0\ -c^2\ 0\ 0\ 0\ 0\ c\} \\
L_3^T &= \{0\ 0\ 0\ 0\ k\ 0\ k\} \\
L_4^T &= \{0\ 0\ 0\ 0\ ck\ 0\ ck\} \\
L_5^T &= \{0\ 0\ 0\ 0\ 0\ 0\ 0\} \\
L_6^T &= \{0\ 0\ 0\ 0\ 0\ 0\ 0\} \\
L_7^T &= \{0\ 0\ 0\ 0\ 0\ 0\ 0\} \\
S_0^T &= \{0\ 0\ 0\ 1\ 0\ 1\ 0\} \\
S_1^T &= \{0\ 0\ 0\ 0\ 0\ 0\ 0\}
\end{aligned}
\tag{A.2}
$$

$$S_2^T = \{0\ 0\ 0\ 3k\ 0\ 3k\ 0\}$$

$$H_0 = \begin{bmatrix} c^2 & 0 & 0 & c & 0 & 0 & 0 \\ 0 & 1 & -c & 0 & 0 & 0 & 0 \\ 0 & -c & c^2 & 0 & 0 & 0 & 0 \\ c & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$H_1 = \begin{bmatrix} 0 & 0 & 0 & c^2 & 0 & c^2 & 0 \\ 0 & 0 & -c^2 & 0 & 0 & 0 & 1 \\ 0 & -c^2 & 2c^3 & 0 & 0 & 0 & -c \\ c^2 & 0 & 0 & 2c & 0 & c & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c^2 & 0 & 0 & c & 0 & 0 & 0 \\ 0 & 1 & -c & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$H_2 = \begin{bmatrix} -3c^4 & 0 & 0 & -2c^3 & 0 & c^3 & 0 \\ 0 & -3c^2 & 2c^3 & 0 & 0 & 0 & c \\ 0 & 2c^3 & 0 & 0 & 0 & 0 & -2c^2 \\ -2c^3 & 0 & 0 & 0 & 0 & 2c^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c^3 & 0 & 0 & 2c^2 & 0 & c^2 & 0 \\ 0 & c & -2c^2 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$
\begin{aligned}
H_3 &= \begin{bmatrix} 2c^5 & 0 & 0 & kc^2 & 0 & -2c^4+kc^2 & 0 \\ 0 & 2c^3 & 0 & 0 & k & 0 & -2c^2+k \\ 0 & 0 & 0 & 0 & -kc & 0 & -kc \\ kc^2 & 0 & 0 & 2kc & 0 & kc & 0 \\ 0 & k & -kc & 0 & 0 & 0 & 0 \\ -2c^4+kc^2 & 0 & 0 & kc & 0 & 2c^3 & 0 \\ 0 & -2c^2+k & -kc & 0 & 0 & 0 & 2c \end{bmatrix} \\
H_4 &= \begin{bmatrix} 0 & 0 & 0 & kc^3 & 0 & kc^3 & 0 \\ 0 & 0 & 0 & 0 & kc & 0 & kc \\ 0 & 0 & 0 & 0 & -2kc^2 & 0 & -2kc^2 \\ kc^3 & 0 & 0 & 4kc^2 & 0 & 3kc^2 & 0 \\ 0 & kc & -2kc^2 & 0 & 0 & 0 & k \\ kc^3 & 0 & 0 & 3kc^2 & 0 & 2kc^2 & 0 \\ 0 & kc & -2kc^2 & 0 & k & 0 & 2k \end{bmatrix} \qquad \text{(A.3)} \\
H_5 &= \begin{bmatrix} 0 & 0 & 0 & -2kc^4 & 0 & -2kc^4 & 0 \\ 0 & 0 & 0 & 0 & -2kc^2 & 0 & -2kc^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2kc^4 & 0 & 0 & 0 & 0 & 2kc^3 & 0 \\ 0 & -2kc^2 & 0 & 0 & 0 & 0 & 2kc \\ -2kc^4 & 0 & 0 & 2kc^3 & 0 & 4kc^3 & 0 \\ 0 & -2kc^2 & 0 & 0 & 2kc & 0 & 4kc \end{bmatrix} \\
H_6 &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & k^2c^2 & 0 & k^2c^2 & 0 \\ 0 & 0 & 0 & 0 & k^2 & 0 & k^2 \\ 0 & 0 & 0 & k^2c^2 & 0 & k^2c^2 & 0 \\ 0 & 0 & 0 & 0 & k^2 & 0 & k^2 \end{bmatrix}
\end{aligned}
$$

$$H_7 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2k^2c^3 & 0 & 2k^2c^3 & 0 \\ 0 & 0 & 0 & 0 & 2k^2c & 0 & 2k^2c \\ 0 & 0 & 0 & 2k^2c^3 & 0 & 2k^2c^3 & 0 \\ 0 & 0 & 0 & 0 & 2k^2c & 0 & 2k^2c \end{bmatrix}$$

Element independent stiffness terms as desribed in Chapter 2:

$$\begin{aligned}\hat{K} &= AL_0L_0^T + D(L_0L_2^T + L_2L_0^T + L_1L_1^T) \\ &+F(L_0L_4^T + L_4L_0^T + L_1L_3^T + L_3L_1^T + L_2L_2^T) \\ &+H(L_2L_4^T + L_4L_2^T + L_3L_3^T) + JL_4L_4^T + AS\ S_0S_0^T \\ &+DS(S_0S_2^T + S_2S_0^T) + FS\ S_2S_2^T\end{aligned} \tag{A.4}$$

$$\begin{aligned}\hat{N}_1 &= A(L_0d^TH_0 + d^TL_0H_0 + H_0dL_0^T) \\ &+D(L_0d^TH_2 + L_2d^TH_0 + L_1d^TH_1 + d^TL_0H_2 + d^TL_2H_0 + d^TL_1H_1 \\ &+H_2dL_0^T + H_0dL_2^T + H_1dL_1^T) \\ &+F(L_0d^TH_4 + L_4d^TH_0 + L_1d^TH_3 + L_3d^TH_1 + L_2d^TH_2 + d^TL_0H_4 \\ &+d^TL_4H_0 + d^TL_1H_3 + d^TL_3H_1 + d^TL_2H_2 + H_4dL_0^T + H_0dL_4^T \\ &+H_3dL_1^T + H_1dL_3^T + H_2dL_2^T) \\ &+H(L_0d^TH_6 + L_1d^TH_5 + L_2d^TH_4 + L_3d^TH_3 + L_4d^TH_2 + d^TL_0H_6 \\ &+d^TL_1H_5 + d^TL_2H_4 + d^TL_3H_3 + d^TL_4H_2 + H_6dL_0^T + H_5dL_1^T \\ &+H_4dL_2^T + H_3dL_3^T + H_2dL_4^T) \\ &+J(L_1d^TH_7 + L_2d^TH_6 + L_3d^TH_5 + L_4d^TH_4 + d^TL_1H_7 + d^TL_2H_6 \\ &+d^TL_3H_5 + d^TL_4H_4 + H_7dL_1^T + H_6dL_2^T + H_5dL_3^T + H_4dL_4^T) \\ &+L(L_3d^TH_7 + L_4d^TH_4 + d^TL_3H_3 + d^TL_4H_4 + H_7dL_3^T + H_4dL_4^T)\end{aligned} \tag{A.5}$$

$$
\begin{aligned}
\hat{N}_2 &= A(H_0dd^TH_0 + \frac{1}{2}d^TH_0dH_0) \\
&+D(\frac{1}{2}H_0dd^TH_2 + \frac{1}{2}H_2dd^TH_0H_1dd^TH_1 + \frac{1}{4}d^TH_0dH_2 + \frac{1}{4}d^TH_2dH_0 \\
&+\frac{1}{2}d^TH_1dH_1) \\
&+F(\frac{1}{2}H_0dd^TH_4 + \frac{1}{2}H_4dd^TH_0 + H_1dd^TH_3 + H_3dd^TH_1 + H_2dd^TH_2 \\
&+\frac{1}{4}d^TH_0dH_4 + \frac{1}{4}d^TH_4dH_0 + \frac{1}{2}d^TH_1dH_3 + \frac{1}{2}d^TH_3dH_1 + \frac{1}{2}d^TH_2dH_2) \\
&+H(\frac{1}{2}H_0dd^TH_6 + \frac{1}{2}H_6dd^TH_0 + H_1dd^TH_5 + H_5dd^TH_1 + H_2dd^TH_4 \\
&+H_4dd^TH_2 + H_3dd^TH_3 + \frac{1}{4}d^TH_0dH_6 + \frac{1}{4}d^TH_6dH_0 + \frac{1}{2}d^TH_1dH_5 \\
&+\frac{1}{2}d^TH_5dH_1 + \frac{1}{2}d^TH_2dH_4 + \frac{1}{2}d^TH_4dH_2 + \frac{1}{2}d^TH_3dH_3) \qquad\qquad (\mathbf{A.6}) \\
&+J(H_1dd^TH_7 + H_7dd^TH_1 + H_2dd^TH_6 + H_6dd^TH_2 + H_3dd^TH_5 \\
&+H_5dd^TH_3 + H_4dd^TH_4 + \frac{1}{2}d^TH_1dH_7 + \frac{1}{2}d^TH_7dH_1 + \frac{1}{2}d^TH_2dH_6 \\
&+\frac{1}{2}d^TH_6dH_2 + \frac{1}{2}d^TH_3dH_5 + \frac{1}{2}d^TH_5dH_3 + \frac{1}{2}d^TH_4dH_4) \\
&+L(H_3dd^TH_7 + H_7dd^TH_3 + H_4dd^TH_6 + H_6dd^TH_4 + H_5dd^TH_5 \\
&+\frac{1}{2}d^TH_3dH_7 + \frac{1}{2}d^TH_7dH_3 + \frac{1}{2}d^TH_4dH_6 + \frac{1}{2}d^TH_6dH_4 + \frac{1}{2}d^TH_5dH_5) \\
&+R(H_5dd^TH_7 + H_7dd^TH_5 + H_6dd^TH_6 + \frac{1}{2}d^TH_5dH_7 + \frac{1}{2}d^TH_7dH_5 \\
&+\frac{1}{2}d^TH_6dH_6) \\
&+T(H_7dd^TH_7 + \frac{1}{2}d^TH_7dH_7)
\end{aligned}
$$

Appendix B. *FORTRAN Program Description*

B.1 Background

Based on the theory presented in Chapter 2, a FORTRAN code was developed. Two-dimensional cylindrical shell codes had previously been developed by Dennis, Tsai and Smith. Dennis (8) developed a displacement control algorithm and Tsai and Palazotto (23) later incorporated the Riks technique. Smith (22) refined Dennis' code to include additional higher-order strain terms and to provide many options so comparisons between theories were possible. Both Dennis and Smith generated stiffness matrix terms in FORTRAN code through the MACSYMA symbolic manipulation program. The present code used simplified versions of Smith's MACSYMA input files to generate the 1-D stiffness matrices. In this way the higher-order strain terms ignored by Dennis were captured.

Several of the subroutines used in the present code are copied from Dennis and Tsai. Those copied subroutines are noted in the subroutine descriptions. In most cases, however, the present code resembles that of Dennis and Tsai only in the numerical solution logic. Enough simplification was possible in transformation fron 2-D to 1-D that the present code, along with comments, is much easier to read and decipher.

B.2 Subroutine Descriptions

This listing is intended to give the reader an overview of the program. Not all subroutines are listed separately. For instance six stiffness term subroutines are listed within the description of *stiff*.

1. *beam*: This is the main program. It simply calls *rinput*, *elast* and either *proces* or *rikspr* depending on whether displacement control or Riks method is chosen.
2. *rinput*: This subroutine reads in the problem data and echos it to the output file in readable format. It prompts the user for input and output file names. It also computes nodal coordinates.
3. *elast*: This subroutine integrates the specified elasticity values through the beam thickness for isotropic or laminate beams.

4. *proces*: This subroutine manages solution by displacement control. It calls *stiff*,*solve*,*converge*, and *postpr* as necessary. It assembles the global stiffness matrix, applies boundary conditions, and increments displacements for each increment. This subroutine contains the *solve* subroutine which solves the symmetric banded equations. *solve* is directly copied from Dennis' program listing. *converge* checks iterations against the specified convergence criteria; it is also copied directly from Dennis.

5. *rikspr*: This subroutine is the Riks equivalent of *proces*. It's more complex than *proces* because the Riks method determining solution increments is much more complex than displacement control. The logic of the Riks solution is copied from Tsai's code.

6. *stiff*: This subroutine computes the elemental stiffness matrices for each element. For an arch, *stiff* calls *beamk* for $\hat{K}$, *beamn1* for $\hat{N}_1$ and *beamn2* for $\hat{N}_2$. For a straight beam,*stiff* calls *sbeamk* for $\hat{K}$, *sbmn1* for $\hat{N}_1$ and *sbmn2* for $\hat{N}_2$. Next *stiff* calls *shape* to compute K, N_1 and N_2. Five–point Gaussian quadrature is used for integration.

7. *shape*: This subroutine is called by *stiff* to compute the shape function matrix at each Gauss point. The shape functions include relavent terms from the inverse Jacobian matrix.

8. *postpr*: This subroutine is called if convergence is achieved for a particular increnment. It computes the resultant forces as requested by the user for the converged displacement values. This force, along with nodal displaacements, is printed to the output file. It also generates an output file named "plot" which contains in column format the displacement at the degree of freedom specified for force computation, displacement at the degree of freedom two less than that specified and the force at the specified degree of freedom. In other words, if degree of freedom 53 is w at an arch crown, then degree of freedom 51, v, appears in the second column and the vertical load appears in the third column. All this makes plotting in other program such as MATLAB very easy. If requested, this subroutine also generatates x, y coordinates for each node at each increment. The computation varies depending whether we have a straight beam or a full arch or if we model half of a symmetric arch.

B.3 Data Input Format

Following are instructions to create a data input file for "BEAM" a FORTRAN code which handles geometrically nonlinear beam and arch problems. The instructions present variable names from the program in the positions required for the program to read them correctly. Variable names appear in *italics* one line at a time. Descriptions of the variables appear on the right. This is standard FORTRAN77 list directed read format.

1. *title*: text string for problem title
2. *linear,isotro,isarch,ishape* :
   (a) *linear* : 0 for a nonlinear problem, 1 for a linear problem.
   (b) *isotro* : 0 for a composite, 1 for isotropic.
   (c) *isarch* : 0 for a straight beam, 1 for a circular arch.
   (d) *ishape* : 1 to compute x,y coordinates for each node each increment for a straight beam or full arch; 2 for a half arch; output goes to file named "bshape"
3. *inctyp,ninc,imax,kupdte,tol* :
   (a) *inctyp* : 1 for displacement control, 2 for Riks method
   (b) *ninc* : number of displacement or load (Riks) increments desired
   (c) *imax* : max number of iterations for an increment
   (d) *kupdte* : not used but fill with 1 or 0
   (e) *tol* : percent convergence desired to stop iterations in an increment; best to use 0.01 or less but you might want to loosen or tighten this for some Riks problems
4. *pincr,eiter,ttpi* : include this line only if inctyp=2 (Riks) and linear=0
   (a) *pincr* : initial load parameter (try 0.1)
   (b) *eiter* : estimate of iterations per increment; this isn't an integer and is valuable to control increment size
   (c) *ttpi* : max load increment or decrement for an iteration; rarely a factor

5. *table(ninc)* : include this line only if inctyp=1 and linear=0 table is an array of ninc values of desired displacements for displacement control
6. *nelem* : number of elements in the mesh
7. *delem(nelem)* : nelem lengths, one for each element
8. *nbndry* : number of nodes with displacement boundary conditions to be specified
9. *nbound(nbndry,5)* : one line for each node with prescribed displacements (nbndry lines); first number in each line is the node number; note each element has 3 nodes but only the end nodes are numbered in this code; i.e. a 10 element mesh has 11 nodes; there's no way to prescribe a displacement at a midnode; next 4 numbers on each line are 1's or 0's, 1=prescribed displacement, 0=free to displace; order of d.o.f.'s is $v, \psi_2, w, w_{,2}$
10. *vbound(ii)* : real prescribed displacements for those d.o.f.'s fixed above in the order from above; for dis- placement control values that appear here are multiplied by the incremental values in table (ninc) in successive increments
11. *ldtyp,distld,ldtyp* : 0 for no distributed load *distld*: intensity of distributed load Note: As of 31Nov92 this option is not available but this line is still necessary
12. *ndload* : skip if *ldtyp*=0; number of elements with dist. load
13. *idload(ndload)* : skip if *ldtyp*=0; numbers of elements with dist. loading
14. *nconc* : number of concentrated loads (and moments); must have at least 1 for Riks technique, 0 if no conc. loads
15. *iconc(nconc)* : skip if *nconc*=0; d.o.f. numbers for concentrated loads; here the middle nodes count; there are 9-d.o.f.'s per element, in order: $v(1)$, $\psi_2(1)$, $w(1)$, $w_{,2}(1)$, $v(3)$, $v(2)$, $\psi_2(2)$, $w(2)$, $w_{,2}(2)$
16. *vconc(nconc)* : skip if *nconc*=0; values of loads at each d.o.f. listed above
17. *ey,nu,ht,width* : include for isotropic material(isotro=1);
    (a) *ey* : Young's modulus,
    (b) *nu* : Poisson's ratio,

   (c) *ht* : thickness,

   (d) *width* : width

18. *e1,e2,g12,nu12,g13,g23,width* : include for composite(isotro=0);

    (a) *e1* : Young's modulus along fibers

    (b) *e2* : Young's modulus transverse to fibers

    (c) *g12* : shear modulus

    (d) *nu12* : Poisson's ratio

    (e) *g13* : 1-3 shear modulus

    (f) *g23* : 2-3 shear modulus

    (g) *width* : width

19. *nplies,pthick* : include for composite(isotro=0)

    (a) *nplies* : number of plies

    (b) *pthick* : ply thickness (one number,same for all plies)

20. *theta(nplies)* : include for composite(isotro=0);ply orientation angles in degrees

21. *rad* : include if isarch=1; arch radius of curvature

22. *nforc* : number of nodal resultant forces to calculate

23. *iforc(nforc)* : include if nforce > 0; nforce d.o.f. numbers of locations for force calculations

24. *nstres* : number of elements where stress is to be calculated as of 31 NOV92 not used but code needs a zero here

25. *istres(nstres)* : skip if nstres=0 *nstres* element numbers for stress calcs

Appendix C. *FORTRAN Code*

```
program beam
c
c See bottom of file for variable and subroutine listing.
c
implicit double precision (a-h,o-z)
character*64 gname
c
c
common/chac/gname,fname
c
common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
common/input/tol,table(250),delem(250),vbound(2500),distld,
     .  vconc(2500),ey,enu,ht,e1,e2,g12,enu12,enu21,g13,g23,pthick,
     .  rad,linear,isotro,isarch,ishape,inctyp,ninc,imax,
     .  nelem,nbndry,nbound(250,5),ldtyp,nconc,iconc(2500),
     .  nplies,nforc,iforc(2500),nstres,istres(250),ibndry(2500),
     .  theta(20),idload(250),coord(251),width,nnod,pincr,eiter,ttpi
c
common/stf/stif(9,9),elp(9),eln(9,9),eld(9)
c
common/proc/gstif(2500,9),gn(2500,9),gf(2500),gd(2500),vperm(2500),
     .   vpres(2500)
c
call rinput
call elast
if(inctyp.eq.1)call proces
if(inctyp.eq.2)call rikspr
c***********************************************************************
c
c VARIABLES FOR BSHELL
c
c fname input file
c gname output file
c ae,de, elasticity terms
c fe,he, elasticity terms
c ej,el, elasticity terms
c re,te, elasticity te
c as,ds, elasticity terms
c fs elasticity term
c ey Young's modulus for isotropic case
c enu Poisson's ratio for isotropic case
c ht thickness of beam for isotropic case
```

```
c e1,e2, laminate material properties
c g12,enu12, "
c enu21,g13, "
c g23 "
c pthick laminate ply thickness
c nplies number of plies in laminate
c theta(20) ply orientation angles
c tol convergence tolerance, percent
c table(250) displacement increment multiplicative
c factors
c delem(250) element lengths
c vbound(2500) values of prescribed displacement boundary
c conditions
c distld distributed load intensity
c vconc(2500) concentrated load values
c rad arch radius of curvature
c linear =1 for linear analysis, =0 for nonlinear
c isotro =1 for isotropic, =0 for laminate
c isarch =1 for arch, =0 for straight beam
c ishape =1 to print x,y coordinates for each node at each increment
c when a full arch is represented output to file 'bshape'
c inclod  =1 to increment load(NA), =0 increment displacement
c ninc total number of displacement increments
c imax maximum number of iterations per increment
c nelem total number of elements in model
c nbndry number of nodes with specified boundary conditions
c nbound(250,5) array of node numbers followed by 1's for
c fixed b.c.'s, zeros for unfixed
c ldtyp =1 for distributed load, =0 no distributed load
c nconc total number of concentrated loads input
c iconc(2500) DOF's for specified loads
c nforc number of forces(including moments)to be solved for
c iforc(2500) DOF's at which to calculate forces
c nstres number of elements for stress calculation
c istres(250) element #'s for stress calculation
c ibndry(2500) DOF numbers for b.c.'s
c idload(250) elements with distributed load
c coord(251) coordinate of the nodes
c width beam or arch width
c nnod number of nodes
c*****************************************************************************
c
c SUBROUTINES FOR BSHELL
c
c rinput reads in and echos input data
```

```
c elast computes elasticity terms
c proces drives the solution algorithm for displacement control
c rikspr  drives the solution algorithm for Riks method
c stiff manages stiffness matrix computations
c shape computes shape function array dsf
c beamk computes constant stiffness array bmk
c beamn1 computes linear stiffness array bmn1
c beamn2 computes quadratic stiffness array bmn2
c sbeamk  computed constant stiffness array for straight beams
c sbmn1 computes linear stiffness array for straight beams
c sbmn2 computes quadratic stiffness array for straight beams
c bndy applies displacement boundary conditions
c solve solves simultaneous equations in banded array format
c converge checks solutions for convergence
c postpr computes nodal loads and sends to output file
c
c
end
c
c
c
subroutine rinput
c
character*64 fname,gname
character*4 title
dimension title(20)
implicit double precision (a-h,o-z)
c
common/chac/gname,fname
common/input/tol,table(250),delem(250),vbound(2500),distld,
     .  vconc(2500),ey,enu,ht,e1,e2,g12,enu12,enu21,g13,g23,pthick,
     .  rad,linear,isotro,isarch,ishape,inctyp,ninc,imax,
     .  nelem,nbndry,nbound(250,5),ldtyp,nconc,iconc(2500),
     .  nplies,nforc,iforc(2500),nstres,istres(250),ibndry(2500),
     .  theta(20),idload(250),coord(251),width,nnod,pincr,eiter,ttpi
c
write(*,1000)
read(*,1005)fname
write(*,1010)
read(*,1005)gname
open(5,file=fname)
open(6,file=gname,status='new')
read(5,1015)title
read(5,*)linear,isotro,isarch,ishape
read(5,*)inctyp,ninc,imax,kupdte,tol
```

```
if(linear.eq.0.and.inctyp.eq.2)read(5,*)pincr,eiter,ttpi
if(linear.eq.0.and.inctyp.eq.1)read(5,*)(table(i),i=1,ninc)
read(5,*)nelem
read(5,*)(delem(i),i=1,nelem)
c
c calculate nodal coordinates
c
nnod=nelem+1
coord(1)=0.0
do 5 ii=2,nnod
   5 coord(ii)=coord(ii-1)+delem(ii-1)
read(5,*)nbndry
do 10 i=1,nbndry
  10    read(5,*)(nbound(i,j),j=1,5)
ifdof=0
c
c ifdof=counter for enumber of fixed dof's
c
do 20 i=1,nbndry
do 20 j=2,5
if(nbound(i,j).eq.0)goto 20
ifdof=ifdof+1
ibndry(ifdof)=(nbound(i,1)-1)*5 + (j-1)
  20    continue
read(5,*)(vbound(i),i=1,ifdof)
read(5,*)ldtyp,distld
if(ldtyp.eq.1)read(5,*)ndload
if(ldtyp.eq.1)read(5,*)(idload(i),i=1,ndload)
read(5,*)nconc
if(nconc.ne.0)read(5,*)(iconc(i),i=1,nconc)
if(nconc.ne.0)read(5,*)(vconc(i),i=1,nconc)
if(isotro.eq.1)read(5,*)ey,enu,ht,width
if(isotro.eq.0)read(5,*)e1,e2,g12,enu12,g13,g23,width
if(isotro.eq.0)read(5,*)nplies,pthick
if(isotro.eq.0)read(5,*)(theta(i),i=1,nplies)
if(isarch.eq.1)read(5,*)rad
read(5,*)nforc
if(nforc.ne.0)read(5,*)(iforc(i),i=1,nforc)
read(5,*)nstres
if(nstres.ne.0)read(5,*)(istres(i),i=1,nstres)
c
c Echo the input to the output file
c
write(6,1015)title
if(isarch.eq.1)write(6,1020)
if(isarch.eq.0)write(6,1025)
if(linear.eq.1)write(6,1030)
if(linear.eq.0)write(6,1035)
if(isotro.eq.1)write(6,1040)
if(isotro.eq.0)write(6,1045)
if(ishape.eq.1)write(6,1050)
if(inctyp.eq.1)write(6,1060)
if(inctyp.eq.2)write(6,1055)
write(6,1065)ninc
write(6,1070)imax
write(6,1075)tol
if(inctyp.eq.2)write(6,1076)pincr,eiter,ttpi
if(inctyp.eq.1)write(6,1078)
if(inctyp.eq.1)write(6,1080)(table(i),i=1,ninc)
write(6,1085)nelem
write(6,1090)
write(6,1095)(coord(i),i=1,nnod)
write(6,1100)
write(6,1105)
do 30 i=1,nbndry
   30 write(6,1110)(nbound(i,j),j=1,5)
write(6,1115)ifdof
write(6,1120)(ibndry(i),i=1,ifdof)
write(6,1095)(vbound(i),i=1,ifdof)
if(ldtyp.eq.1)write(6,1125)distld
if(ldtyp.eq.1)write(6,1130)(idload(i),i=1,ndload)
if(nconc.ne.0)write(6,1135)
if(nconc.ne.0)write(6,1120)(iconc(i),i=1,nconc)
if(nconc.ne.0)write(6,1095)(vconc(i),i=1,nconc)
if(isotro.eq.1)write(6,1140)ey,enu,ht,width
if(isotro.eq.0)write(6,1145)e1,e2,g12,enu12,g13,g23,width
if(isotro.eq.0)write(6,1150)nplies,pthick
if(isotro.eq.0)write(6,1155)(theta(i),i=1,nplies)
if(isarch.eq.1)write(6,1160)rad
write(6,1165)(iforc(i),i=1,nforc)
write(6,1170)(istres(i),i=1,nstres)
close(5)
c close(6)
c
c
c F  O  R  M  A  T  S
c
 1000 format('Enter your input file name.')
 1005 format(A)
 1010 format('Enter your output file name.')
 1015 format(20a4)
 1020 format(/,1x,'Element type: arch')
 1025 format(/,1x,'Element type: straight beam')
 1030   format(/,1x,'Analysis type: linear')
 1035 format(/,1x,'Analysis type: nonlinear')
 1040 format(/,1x,'Material type: isotropic')
 1045 format(/,1x,'Material type: laminate')
 1050 format(/,1x,'Printout of nodal x,y coordinates requested')
 1055 format(/,1x,'Riks method specified')
 1060 format(/,1x,'Displacement control method specified')
 1065 format(/,1x,'Increments specified:',2x,i3)
 1070 format(/,1x,'Maximum iterations specified:',2x,i3)
 1075 format(/,1x,'Percent convergence tolerance:',2x,d12.5)
 1076 format(/,1x,'pincr=',2x,d12.5,2x,'eiter=',2x,d12.5,2x,
     .     'ttpi=',2x,d12.5)
 1078 format(/,1x,'Displacement Increment Table')
 1080 format(8(2x,d12.5))
 1085 format(/,1x,'Number of elements:',2x,i3)
 1090 format(/,1x,'Nodal Coordinates:')
 1095 format(8(2x,d12.5))
 1100 format(/,1x,'DISPLACEMENT BOUNDARY CONDITIONS, 1=PRESCRIBED,
     X0=FREE')
 1105 format(/,4X,'NODE    V   PSI-S   W   W-S  ')
 1110 format(4x,i4,1x,4(i3,2x))
 1115 format(/,1x,'NUMBER OF PRESCRIBED DISPLACEMENTS:',
     .  i5,/,1x,'SPECIFIED DISPLACEMENT DOF AND THIER
     .  VALUES FOLLOW:')
 1120 format(16i5)

 1125 format(/,1x,'Distributed Load Intensity:',2x,d12.5)
 1130 format(/,1x,'Elements with distributed load:',/,1x,16i5)
 1135 format(/,1x,'DOF and specified concentrated loadsfollow:')
 1140 format(/,1x,'Isotropic material properties ey, enu, ht, width:'
     .  ,/,1x,4d12.5)
 1145 format(/,1x,'Composite material properties e1, e2, g12, enu12,
     .  g13,g23, width:',/,1x,7d12.5)
 1150 format(/,1x,'Number of plies:',2x,i3,2x,'Ply thickness:',2x,
     .  d12.5)
 1155 format(/,1x,'Ply orientation angles:',/,1x,8(2x,d12.5))
 1160 format(/,1x,'Radius of curvature:',2x,d12.5)
 1165 format(/,1x,'DOFs for equivalent load calculation:',/,
     .  1x,16i5)
 1170 format(/,1x,'Elements for stress calculation:',/,1x,16i5)
 1175 format(/,1x,i5)
return
```

```
end
c
c
c
subroutine elast
c
implicit double precision (a-h,o-z)
c
dimension qbar(3,3),rtheta(20)
c
c
character*64 gname
common/chac/gname,fname
common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
common/input/tol,table(250),delem(250),vbound(2500),distld,
     .  vconc(2500),ey,enu,ht,e1,e2,g12,enu12,enu21,g13,g23,pthick,
     .  rad,linear,isotro,isarch,ishape,inctyp,ninc,imax,
     .  nelem,nbndry,nbound(250,5),ldtyp,nconc,iconc(2500),
     .  nplies,nforc,iforc(2500),nstres,istres(250),ibndry(2500),
     .  theta(20),idload(250),coord(251),width,nnod,pincr,eiter,ttpi
c
c
c Isotropic case
c
c write(6,1000)e1,e2,g12,enu12,enu21,g13,g23,pthick
if(isotro.eq.0)goto 100
gs=ey/(2*(1+enu))
       denom=1.-enu**2
       q11=ey/denom
q12=enu*ey/denom
q22=q11
q2hat=q22-(q12**2/q11)
qs4=gs
ae=q2hat*ht
       de=q2hat*ht**3/(3*2.**2)
       fe=q2hat*ht**5/(5*2.**4)
       he=q2hat*ht**7/(7*2.**6)
       ej=q2hat*ht**9/(9*2.**8)
       el=q2hat*ht**11/(11*2.**10)
       re=q2hat*ht**13/(13*2.**12)
    te=q2hat*ht**15/(15*2.**14)
as=qs4*ht
ds=qs4*ht**3/(3*2.**2)
fs=qs4*ht**5/(5*2.**4)
goto 200
```

```
c
c Laminate case
c
 100 ht=pthick*nplies
       enu21=e2*enu12/e1
       denom=1.-enu12*enu21
       q11=e1/denom
       q12=enu12*e2/denom
       q22=e2/denom
c
c**************************************************
c   calculate the elasticity matrices            *
c                                                    *
c   remem that the z axis points down,           *
c   however, the first ply is the top ply, ie,   *
c   the ply with the most negative z !!!         *
c**************************************************
c
c initialize elasticity terms
c
ae=0.
       de=0.
       fe=0.
       he=0.
       ej=0.
       el=0.
       re=0.
    te=0.
as=0.
ds=0.
fs=0.
       do 45 ii=1,nplies
  45   rtheta(ii)=theta(ii)*3.14159265/180.
       do 50 kk=1,nplies
       qbar(1,1)=q11*(cos(rtheta(kk))**4)+2*q12*(sin(rtheta(kk))**2)*
     .   (cos(rtheta(kk))**2)+q22*(sin(rtheta(kk))**4)
       qbar(1,2)=(q11+q22)*(sin(rtheta(kk))**2)*(cos(rtheta(kk))**2)+
     .   q12*(sin(rtheta(kk))**4+cos(rtheta(kk))**4)
       qbar(2,2)=q11*(sin(rtheta(kk))**4)+2*q12*(sin(rtheta(kk))**2)*
     .   (cos(rtheta(kk))**2)+q22*cos(rtheta(kk))**4
       qs4=g13*dcos(rtheta(kk))**2+g23*dsin(rtheta(kk))**2
q2hat=qbar(2,2)-(qbar(1,2)**2/qbar(1,1))
zl=(kk*1. - nplies*.5)*pthick
       zu=zl-pthick
ae=ae + q2hat*pthick
```

```
de=de + q2hat*(zl**3-zu**3)/3.
       fe=fe + q2hat*(zl**5-zu**5)/5.
       he=he + q2hat*(zl**7-zu**7)/7.
       ej=ej + q2hat*(zl**9-zu**9)/9.
        el=el + q2hat*(zl**11-zu**11)/11.
        re=re + q2hat*(zl**13-zu**13)/13.
       te=te + q2hat*(zl**15-zu**15)/15.
as=as+qs4*pthick
       ds=ds+qs4*(zl**3-zu**3)/3.
     fs=fs+qs4*(zl**5-zu**5)/5.
  50 continue
c 200 open(6,fiel=gname,status='old')
  200 write(6,1000)ae,de,fe,he,ej,el,re,te,as,ds,fs
c close(6)
 1000 format(/,1x,'Elasticity terms:',/,1x,8(2x,d12.5))
    return
end
c
c
c
subroutine proces
c
implicit double precision (a-h,o-z)
c
character*64 gname
c
c
common/chac/gname,fname
c
common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
c
common/input/tol,table(250),delem(250),vbound(2500),distld,
     .  vconc(2500),ey,enu,ht,e1,e2,g12,enu12,enu21,g13,g23,pthick,
     .  rad,linear,isotro,isarch,ishape,inctyp,ninc,imax,
     .  nelem,nbndry,nbound(250,5),ldtyp,nconc,iconc(2500),
     .  nplies,nforc,iforc(2500),nstres,istres(250),ibndry(2500),
     .  theta(20),idload(250),coord(251),width,nnod,pincr,eiter,ttpi
common/stf/stif(9,9),elp(9),eln(9,9),eld(9)
c
common/proc/gstif(2500,9),gn(2500,9),gf(2500),gd(2500),vperm(2500),
     .   vpres(2500)
c
ndof=nnod*4+nelem
ncount=1
icount=1
```

```
do 1 ii=1,ndof
  1 gd(ii)=0.0d0
do 2 ii=1,nbndry*5
vpres(ii)=0.0d0
  2 vperm(ii)=vbound(ii)
c
c start new increment or iteration/
c zero out global stiffness matrices and global force
c vector
c
  3 do 5 ii=1,ndof
gf(ii)=0.0d0
do 5 jj=1,9
gstif(ii,jj)=0.0d0
  5 gn(ii,jj)=0.0d0
kcall=0
c
c increment prescribed displacement for displacement control
c
if(linear.eq.1)goto 9
if(icount.ne.1)goto 9
do 7 ii=1,nbndry*5
if(ncount.eq.1)vbound(ii)=vperm(ii)*table(1)
   7 if(ncount.gt.1)vbound(ii)=vperm(ii)*(table(ncount)-
     . table(ncount-1))
c
c loop over all elements for stiffness and forces
c
   9 do 30 ielem=1,nelem
do 10 ii=1,9
  10 eld(ii)=gd(ii+(ielem-1)*5)
c
kcall=kcall+1
call stiff(ielem,icount,ncount,kcall)
c
c Assemble global stiffness array, gstif, global equilibrium
c stiffness, gn, in banded form. Half-bandwidth=9.  Also
c assemble global force vector, gf.
c
nr=(ielem-1)*5 + 1
do 30 jj=0,8
gf(nr+jj)=gf(nr+jj)+elp(jj+1)
do 30 kk=1,9-jj
gstif(nr+jj,kk)=gstif(nr+jj,kk)+stif(jj+1,kk+jj)
if(linear.eq.1)goto 30
if(icount.eq.1 .and. ncount.eq.1)goto 30
   gn(nr+jj,kk)=gn(nr+jj,kk)+eln(jj+1,kk+jj)
  30 continue
c
c impose force boundary conditions
c at this point, gf=R
c
if(nconc.eq.0)goto 45
do 40 ii=1,nconc
nb=iconc(ii)
  40 gf(nb)=gf(nb)+vconc(ii)
  45 continue
c
c calculate the residual force vector for nonlinear
c analysis. -[gn]*{gd}+R=-[k+n1/2+n2/3]*{q}+R=gf
c
       if(icount.eq.1)goto 65
       do 60 ii=1,ndof
       add=0.
       do 50 kk=1,ii-1
       if(ii-kk+1 .gt. 9)goto 50
       add=add+gn(kk,ii-kk+1)*gd(kk)
   50  continue
       res=0.
       do 55 jj=1,9
       if(jj+ii-1 .gt. ndof)goto 55
       res=res + gn(ii,jj)*gd(jj+ii-1)
   55  continue
c
c add to existing gf which already contains R
c
     gf(ii)=gf(ii)-res-add
   60 continue
   65 continue
c
c impose displacement boundary conditions
c
if(icount.eq.1)call bndy(ndof,gstif,gf,nbndry,ibndry,vbound)
if(icount.gt.1)call bndy(ndof,gstif,gf,nbndry,ibndry,vpres)
c
c solve system of equations, result in gf
c
call solve(ndof,gstif,gf,0,detm,detm1)
c
c update total displacement vector gd
```

```
c
do 70 ii=1,ndof
   70 gd(ii)=gd(ii)+gf(ii)
if(linear.eq.1)goto 80
call converge(ndof,ncon,icount,tol,imax)
c
c if no convergence (ncon=0) start next iteration
c
if(ncon.eq.0)goto 3
   80  continue
if(ncon.eq.1 .and. ncount.le.ninc)then
  call postpr(icount,ncount,kcall,ndof)
  if(ncount.eq.ninc)stop
  ncount=ncount+1
  icount=1
  goto 3
endif
return
end
c
      subroutine bndy(ndof,s,sl,ndum,idum,vdum)
c
c     ..................................................................
c     subroutine used to impose boundary conditions on banded equations
c     ..................................................................
c
      implicit double precision (a-h,o-z)
      dimension s(2500,9),sl(2500)
      dimension idum(ndum*5),vdum(ndum*5)
      do 300 nb = 1, ndum*5
      ie = idum(nb)
      sval = vdum(nb)
      it=8
      i=ie-9
      do 100 ii=1,it
      i=i+1
      if (i .lt. 1)  go to 100
      j=ie-i+1
      sl(i)=sl(i)-s(i,j)*sval
      s(i,j)=0.0
  100 continue
      s(ie,1)=1.0
      sl(ie)=sval
      i=ie
      do 200 ii=2,9
      i=i+1
      if (i .gt. ndof)  go to 200
      sl(i)=sl(i)-s(ie,ii)*sval
      s(ie,ii)=0.0
  200 continue
  300 continue
      return
      end
c
c
c
c
      subroutine converge(ndof,ncon,icount,tol,imax)
c............................................................
c checks for convergnece using global displacement criterion
c............................................................
      implicit double precision (a-h,o-z)
common/proc/gstif(2500,9),gn(2500,9),gf(2500),gd(2500),vperm(2500),
     .   vpres(2500)
c
      rcurr=0.
      do 10 m=1,ndof
  10  rcurr=rcurr + gd(m)*gd(m)
      if(icount.eq.1)rinit=rcurr
      if(icount.eq.1)ncon=0
      if(icount.eq.1)goto 20
c new criteria
      ratio=100. * abs(sqrt(rcurr)-sqrt(pvalue))/sqrt(rinit)
      if(ratio.le.tol)ncon=1
  20  pvalue=rcurr
      write(*,100)ncon,ratio,rinit,rcurr
  100 format(1x,'ncon= ',i3,3x,'ratio= ',d14.6,' rinit= ',d14.6,
     x          ' rcurr= ',d14.6)
      if(icount.eq.imax)write(6,200)
      if(icount.eq.imax)stop
  200 format(1x,'icount equals imax')
      if(ncon.eq.0)icount=icount+1
      return
      end
c
c
c
subroutine rikspr
c
implicit double precision (a-h,o-z)
```

```
c
character*64 gname
c
c
common/chac/gname,fname
c
common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
c
common/input/tol,table(250),delem(250),vbound(2500),distld,
      .  vconc(2500),ey,enu,ht,e1,e2,g12,enu12,enu21,g13,g23,pthick,
      .  rad,linear,isotro,isarch,ishape,inctyp,ninc,imax,
      .  nelem,nbndry,nbound(250,5),ldtyp,nconc,iconc(2500),
      .  nplies,nforc,iforc(2500),nstres,istres(250),ibndry(2500),
      .  theta(20),idload(250),coord(251),width,nnod,pincr,eiter,ttpi
common/stf/stif(9,9),elp(9),eln(9,9),eld(9)
c
common/proc/gstif(2500,9),gn(2500,9),gf(2500),gd(2500),vperm(2500),
      .   vpres(2500)
c
dimension gld(2500),gld0(2500),gld1(2500),gdis(2500),gsti00(2500,9),
      .   gf0(2500),gd00(2500)
ndof=nnod*4+nelem
ncount=1
icount=1
iicut=0
do 1 ii=1,ndof
  1 gd(ii)=0.0d0
do 2 ii=1,nbndry*5
vpres(ii)=0.0d0
  2 vperm(ii)=vbound(ii)
c
c start new increment or iteration/
c zero out global stiffness matrices and global force
c vector
c
tpincr=0.0
if(ncount.eq.1)goto 2993
c
c start new increment
c
  3 if(iicut.eq.0)dss=dss*eiter/icount
 2993 icount=1
   do 2992 ii=1,ndof
gld0(ii)=0.0d0
 2992 gd00(ii)=gd(ii)
```

```
c
c start new iteration
c
  4 do 5 ii=1,ndof
gf0(ii)=0.0d0
do 5 jj=1,9
gstif(ii,jj)=0.0d0
  5 gn(ii,jj)=0.0d0
kcall=0
c
c increment prescribed displacement for displacement control
c
c if(linear.eq.1)goto 9
c if(icount.ne.1)goto 9
c do 7 ii=1,nbndry*5
c if(ncount.eq.1.and.iicut.eq.0)vbound(ii)=vperm(ii)*table(1)
c  7 if(ncount.gt.1.or.iicut.gt.0)vbound(ii)=vperm(ii)*(table(ncount)-
c     . table(ncount-1))
c
c loop over all elements for stiffness and forces
c
   9 do 30 ielem=1,nelem
do 10 ii=1,9
  10 eld(ii)=gd(ii+(ielem-1)*5)
c
kcall=kcall+1
call stiff(ielem,icount,ncount,kcall)
c
c Assemble global stiffness array, gstif, global equilibrium
c stiffness, gn, in banded form. Half-bandwidth=9.  Also
c assemble global force vector, gf.
c
nr=(ielem-1)*5 + 1
do 30 jj=0,8
gf0(nr+jj)=gf0(nr+jj)+elp(jj+1)
do 30 kk=1,9-jj
gstif(nr+jj,kk)=gstif(nr+jj,kk)+stif(jj+1,kk+jj)
if(linear.eq.1)goto 30
if(icount.eq.1 .and. ncount.eq.1 .and.iicut.eq.0)goto 30
   gn(nr+jj,kk)=gn(nr+jj,kk)+eln(jj+1,kk+jj)
  30 continue
c
c impose force boundary conditions
c at this point, gf=R
c
```

```
if(nconc.eq.0)goto 45
do 40 ii=1,nconc
nb=iconc(ii)
  40 gf0(nb)=gf0(nb)+vconc(ii)
  45 continue
do 47 ii=1,ndof
  47 gf(ii)=gf0(ii)
do 48 ii=1,ndof
do 48 jj=1,9
  48 gsti00(ii,jj)=gstif(ii,jj)
call bndy(ndof,gstif,gf,nbndry,ibndry,vbound)
c
call solve(ndof,gstif,gf,0,detm,detml)
dss0=0.0
do 49 ii=1,ndof
gdis(ii)=gf(ii)
  49 dss0=dss0+gf(ii)*gf(ii)
if(icount.ne.1) go to 144
       detm1=detm2
       detm2=detm
       if(ncount.eq.1.and.detm.lt.0.0 .and.iicut.eq.0) pincr=-pincr
       if(ncount.eq.1.and.iicut.eq.0) dss=pincr*dsqrt(dss0)
       if(ncount.ne.1.or.iicut.gt.0) pincr= dss/dsqrt(dss0)*detm*detm1
     .                      *pincr1/dabs(pincr1)
c
c attempt at offloading at bifurcation points
c
c if(iicut.eq.1)pincr=-pincr
c
       pincr1=pincr

       prs=0.0
       do 142 ii=1,ndof
  142  prs=prs+gf0(ii)*gld(ii)
      stifpa=pincr*prs
do 143 ii=1,ndof
  143 gld(ii)=pincr*gdis(ii)
  144 continue
c
c calculate the residual force vector for nonlinear
c analysis. -[gn]*{gd}+R=-[k+n1/2+n2/3]*{q}+R=gf
c

if(icount.eq.1)goto 69
       do 60 ii=1,ndof
```

```
        add=0.
        do 50 kk=1,ii-1
        if(ii-kk+1 .gt. 9)goto 50
        add=add+gn(kk,ii-kk+1)*gd(kk)
   50   continue
        res=0.
       do 55 jj=1,9
       if(jj+ii-1 .gt. ndof)goto 55
        res=res + gn(ii,jj)*gd(jj+ii-1)
   55   continue
c
c add to existing gf which already contains R
c
      gf(ii)=gf0(ii)*(pincr+tpincr)-res-add
   60 continue
   65 continue
c
c impose displacement boundary conditions
c
call bndy(ndof,gsti00,gf,nbndry,ibndry,vbound)
c if(icount.gt.1)call bndy(ndof,gstif,gf,nbndry,ibndry,vpres)
c
c solve system of equations, result in gf
c
call solve(ndof,gstif,gf,1,detm,detml)
c
c through line 69 copied from Tsai's program
c
      a1=dss0
      a2=0.0
      a3=0.0
      do 147 ii=1,ndof
      a2=a2+(gld(ii)+gf(ii))*gdis(ii)
  147 a3=a3+gf(ii)*(2.0*gld(ii)+gf(ii))
      d12=a2*a2-a1*a3
c     write(6,*) d12,a1,a2,a3
c
c
      if(d12.lt.0.0)then
c
c deal with complex roots  by cutting the search
c radius (dss) in half
c
        do 2991 ii=1,ndof
 2991 gd(ii)=gd00(ii)
```

```
iicut=iicut + 1
if(iicut.gt.10)then
write(6,3000)
stop
endif
dss=dss/2.0
goto 3
endif
      iicut=0
      dpinc1=(-a2+dsqrt(d12))/a1
      dpinc2=(-a2-dsqrt(d12))/a1
      theta1=0.0
      theta2=0.0
      do 148 ii=1,ndof
      gld0(ii)=gld(ii)
      gld(ii)=gld(ii)+gf(ii)+dpinc1*gdis(ii)
      gld1(ii)=gld0(ii)+gf(ii)+dpinc2*gdis(ii)
      theta1=theta1+gld0(ii)*gld(ii)
      theta2=theta2+gld0(ii)*gld1(ii)
  148 continue
c      write(6,*) theta1,theta2
      thet12=theta1*theta2
      if(thet12.gt.0.0) go to 149
      dpincr=dpinc1
      if(theta2.gt.0.0) call chsign(gld,gld1,dpincr,dpinc2,ndof)
      go to 150
  149 dpib=-a3/(a2*2.0)
      dpin1=dabs(dpib-dpinc1)
      dpin2=dabs(dpib-dpinc2)
      dpincr=dpinc1
      if(dpin2.lt.dpin1) call chsign(gld,gld1,dpincr,dpinc2,ndof)
  150 pincr=pincr+dpincr
   69 continue
c
c update total displacement vector gd
c
do 70 ii=1,ndof
   70 gd(ii)=gd(ii)+gld(ii)-gld0(ii)
if(linear.eq.1)goto 80
call converge(ndof,ncon,icount,tol,imax)
c
c if no convergence (ncon=0) start next iteration
c
if(ncon.eq.0)goto 4
   80  continue
```

```
if(ncon.eq.1 .and. ncount.le.ninc)then
  call postpr(icount,ncount,kcall,ndof)
  if(ncount.eq.ninc)stop
  ncount=ncount+1
  tpincr=tpincr+pincr
  goto 3
endif
 3000    format(1x,'More than 10 consecutive imaginary roots')
 3010 format(/,1x,i2)
return
end
c
c
c
      subroutine solve(ndof,band,rhs,ires,detm,detml)
c     ...........................................................
c     solve a banded symmetric system of equations
c     ...........................................................
c
      implicit double precision (a-h,o-z)
      dimension band(2500,9),rhs(2500)
      meqns=ndof-1
      if(ires.gt.0)goto 90
      do 500 npiv=1,meqns
c      print*,'npiv= ',npiv
      npivot=npiv+1
      lstsub=npiv+9-1
      if(lstsub.gt.ndof) lstsub=ndof
      do 400 nrow=npivot,lstsub
c     invert rows and columns for row factor
      ncol=nrow-npiv+1
      factor=band(npiv,ncol)/band(npiv,1)
      do 200 ncol=nrow,lstsub
      icol=ncol-nrow+1
      jcol=ncol-npiv+1
  200 band(nrow,icol)=band(nrow,icol)-factor*band(npiv,jcol)
  400 rhs(nrow)=rhs(nrow)-factor*rhs(npiv)
  500 continue
      detm=1.0
      detml=0.0
      do 600 ii=1,ndof
      detml=detml+dlog10(dabs(band(ii,1)))
  600 detm=detm*band(ii,1)/dabs(band(ii,1))
      go to 101
   90 do 100 npiv=1,meqns
```

```
      npivot=npiv+1
      lstsub=npiv+9-1
      if(lstsub.gt.ndof) lstsub=ndof
      do 110 nrow=npivot,lstsub
      ncol=nrow-npiv+1
      factor=band(npiv,ncol)/band(npiv,1)
  110 rhs(nrow)=rhs(nrow)-factor*rhs(npiv)
 100  continue
c     back substitution
  101 do 800 ijk=2,ndof
      npiv=ndof-ijk+2
      rhs(npiv)=rhs(npiv)/band(npiv,1)
      lstsub=npiv-9+1
      if(lstsub.lt.1) lstsub=1
      npivot=npiv-1
      do 700 jki=lstsub,npivot
      nrow=npivot-jki+lstsub
      ncol=npiv-nrow+1
      factor=band(nrow,ncol)
  700 rhs(nrow)=rhs(nrow)-factor*rhs(npiv)
  800 continue
      rhs(1)=rhs(1)/band(1,1)
      return
      end
c
c
c
      subroutine chsign(gld,gld1,dpincr,dpinc2,ndof)
      implicit double precision (a-h,o-z)
      dimension gld(2500),gld1(2500)
      do 100 i=1,ndof
 100  gld(i)=gld1(i)
      dpincr=dpinc2
      return
      end
c
c
c
subroutine stiff(ielem,icount,ncount,kcall)
c
implicit double precision (a-h,o-z)
character*64 gname
common/chac/gname,fname
c
common/input/tol,table(250),delem(250),vbound(2500),distld,
     .  vconc(2500),ey,enu,ht,e1,e2,g12,enu12,enu21,g13,g23,pthick,
     .  rad,linear,isotro,isarch,ishape,inctyp,ninc,imax,
     .  nelem,nbndry,nbound(250,5),ldtyp,nconc,iconc(2500),
     .  nplies,nforc,iforc(2500),nstres,istres(250),ibndry(2500),
     .  theta(20),idload(250),coord(251),width,nnod,pincr,eiter,ttpi
c
      common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
c
      common/stf/stif(9,9),elp(9),eln(9,9),eld(9)
c
      common/shp/dsf(7,9)
c
      dimension bmk(7,7),bmn1(7,7),bmn2(7,7),
     .  gauss4(4),wt4(4),gauss7(7),wt7(7),q(7),dsftr(9,7),
     . pkt(7,7),pkn(7,7),pktd(7,9),pknd(7,9),gauss5(5),wt5(5)
c
      data gauss4/0.8611363115d0,0.3399810435d0,-0.3399810435d0,
     .   -0.8611363115d0/
      data wt4/0.3478548451d0,0.6521451548d0,0.6521451548d0,
     .   0.3478548451d0/
      data gauss5/0.9061798459d0,0.5384693101d0,0.0d0,-0.5384693101d0,
     .   -0.9061798459d0/
      data wt5/0.2369268851d0,0.4786286705d0,0.5688888889d0,
     .   0.4786286705d0,0.2369268851d0/
      data gauss7/0.9491079123d0,0.7415311856d0,0.4058451513d0,
     .   0.0d0,-0.4058451513d0,-0.7415311856d0,-0.9491079123d0/
      data wt7/0.1294849662d0,0.2797053915d0,0.3818300505d0,
     .   0.4179591836d0,0.3818300505d0,0.2797053915d0,0.1294849662d0/
c
c initialize stiffness arrays and load array
c
      do 10 ii=1,9
      elp(ii)=0.0
      do 10 jj=1,9
      stif(ii,jj)=0.0
   10 eln(ii,jj)=0.0
c
c set number of gauss points for interpolation
c
      ngp=5
      if(ncount.eq.1 .and. icount.eq.1)ngp=4
      if(linear.eq.1)ngp=4
c
      ek1=-4./(3.*ht**2)
      p1=1./rad
```

```
if(ncount.eq.1 .and. icount.eq.1 .and. kcall.eq.1 .and. isarch.eq.1)
     .    call beamk(bmk,ek1,p1)
if(ncount.eq.1 .and. icount.eq.1 .and. kcall.eq.1 .and. isarch.eq.0)
     .    call sbeamk(bmk,ek1)
c
c loop over gauss points
c
do 100 ii=1,ngp
if(ngp.eq.4)eta=gauss4(ii)
if(ngp.eq.5)eta=gauss5(ii)
if(ngp.eq.7)eta=gauss7(ii)
   call shape(eta,ielem,aa)
c
c multiply element displacement vector, eld (this is 'q'
c in the thesis) by the shape function matrix, dsf, to get
c the displacement gradient vector(d(s) in thesis, q here)
c
do 20 kk=1,7
   20 q(kk)=0.0
c
do 30 jj=1,7
do 30 kk=1,9
   30 q(jj)=q(jj)+dsf(jj,kk)*eld(kk)
c
c
c initialize bmn1, bmn2
c
do 35 kk=1,7
do 35 jj=1,7
bmn1(jj,kk)=0.0d0
  35 bmn2(jj,kk)=0.0d0
c
c skip bmn1 and bmn2 comps first time through
c
if(icount .eq. 1 .and. ncount .eq. 1)goto 37
c
if(isarch.eq.1)call beamn1(q,bmn1,ek1,p1)
if(isarch.eq.1)call beamn2(q,bmn2,ek1,p1)
if(isarch.eq.0)call sbmn1(q,bmn1,ek1)
if(isarch.eq.0)call sbmn2(q,bmn2,ek1)
  37    continue
c
c transpose the shape function matrix
c
do 40 jj=1,7
do 40 kk=1,9
   40 dsftr(kk,jj)=dsf(jj,kk)
c
c create element independent incremental stiffness array,
c pkt, and element ind. equilibrium stiffness array, pkn
c
do 50 jj=1,7
do 50 kk=1,7
pkt(jj,kk)=bmk(jj,kk)+bmn1(jj,kk)+bmn2(jj,kk)
   50 pkn(jj,kk)=bmk(jj,kk)+bmn1(jj,kk)/2.+bmn2(jj,kk)/3.
c
c post-multiply each array by the shape function matrix
c
do 60 jj=1,7
do 60 kk=1,9
pktd(jj,kk)=0.0
pknd(jj,kk)=0.0
do 60 ll=1,7
pktd(jj,kk)=pktd(jj,kk) + aa*pkt(jj,ll)*dsf(ll,kk)
   60 pknd(jj,kk)=pknd(jj,kk) + aa*pkn(jj,ll)*dsf(ll,kk)
c
c Finally, pre-multiply these new arrays by the transpose
c of the shape function matrix to get the element incremental
c stiffness, stif, and element equilibrium stiffness, eln.
c Also multiply by the weighting factor for this particular
c gauss point.  Note that these arrays are zeroed outside the
c loop over the gauss points since they accumulate (integrate)
c data over all the gauss points.
c
if(ngp.eq.4)wt=wt4(ii)
if(ngp.eq.5)wt=wt5(ii)
if(ngp.eq.7)wt=wt7(ii)
do 70 jj=1,9
do 70 kk=1,9
do 70 ll=1,7
stif(jj,kk)=stif(jj,kk)+wt*width*dsftr(jj,ll)*pktd(ll,kk)
   70 eln(jj,kk)=eln(jj,kk)+wt*width*dsftr(jj,ll)*pknd(ll,kk)
  100   continue
c write(6,1010)ielem
c write(6,1005)
c do 900 ii=1,9
c 900 write(6,1000) (stif(ii,jj),jj=1,9)
 1000 format(9(2x,d12.5))
 1005 format(/,'stif')
 1010 format(i4)
```

```
return
end
c
c
c
subroutine shape(eta,ielem,aa)
c
implicit double precision (a-h,o-z)
c
c
common/shp/dsf(7,9)
c
c
common/input/tol,table(250),delem(250),vbound(2500),distld,
     .  vconc(2500),ey,enu,ht,e1,e2,g12,enu12,enu21,g13,g23,pthick,
     .  rad,linear,isotro,isarch,ishape,inctyp,ninc,imax,
     .  nelem,nbndry,nbound(250,5),ldtyp,nconc,iconc(2500),
     .  nplies,nforc,iforc(2500),nstres,istres(250),ibndry(2500),
     .  theta(20),idload(250),coord(251),width,nnod,pincr,eiter,ttpi
c
c initialize shape function matrix
c
do 10 ii=1,7
do 10 jj=1,9
   10 dsf(ii,jj)=0.0
c
aa=(coord(ielem+1)-coord(ielem))*0.5
c
c enter values into dsf
c these include jacobian terms
c
c
c Q1,Q3,Q2 and derivatives
c
dsf(1,1)=0.5*(eta**2-eta)
dsf(1,5)=1.0-eta**2
dsf(1,6)=0.5*(eta**2+eta)
dsf(2,1)=(eta-0.5)/aa
dsf(2,5)=-2.0*eta/aa
dsf(2,6)=(eta+0.5)/aa
dsf(3,3)=0.25*(2.0-3.0*eta+eta**3)
dsf(3,4)=0.25*aa*(1.0-eta-eta**2+eta**3)
dsf(3,8)=0.25*(2.0+3.0*eta-eta**3)
dsf(3,9)=0.25*aa*(-1.0-eta+eta**2+eta**3)
dsf(4,3)=0.25*(-3.0+3.0*eta**2)/aa
dsf(4,4)=0.25*(-1.0-2.0*eta+3.0*eta**2)
dsf(4,8)=0.25*(3.0-3.0*eta**2)/aa
dsf(4,9)=0.25*(-1.0+2.0*eta+3.0*eta**2)
dsf(5,3)=0.25*6.0*eta/aa**2
dsf(5,4)=0.25*(-2.0+6.0*eta)/aa
dsf(5,8)=-0.25*6.0*eta/aa**2
dsf(5,9)=0.25*(2.0+6.0*eta)/aa
dsf(6,2)=0.5*(1.0-eta)
dsf(6,7)=0.5*(1.0+eta)
dsf(7,2)=-0.5/aa
dsf(7,7)=0.5/aa
c
c temporary print
c
c do 100 ii=1,7
c 100 write(6,1000) (dsf(ii,jj),jj=1,9)
c 1000 format (9(2x,d12.5))
return
end
c
c
c
subroutine postpr(icount,ncount,kcall,ndof)
c
implicit double precision (a-h,o-z)
c
character*64 gname
c
c
common/chac/gname,fname
c
common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
c
common/input/tol,table(250),delem(250),vbound(2500),distld,
     . vconc(2500),ey,enu,ht,e1,e2,g12,enu12,enu21,g13,g23,pthick,
     . rad,linear,isotro,isarch,ishape,inctyp,ninc,imax,
     . nelem,nbndry,nbound(250,5),ldtyp,nconc,iconc(2500),
     . nplies,nforc,iforc(2500),nstres,istres(250),ibndry(2500),
     . theta(20),idload(250),coord(251),width,nnod,pincr,eiter,ttpi
c
common/stf/stif(9,9),elp(9),eln(9,9),eld(9)
c
common/proc/gstif(2500,9),gn(2500,9),gf(2500),gd(2500),vperm(2500),
     .  vpres(2500)
c
```

```
dimension vforc(2500),xcoord(251),ycoord(251)
pi=3.14159
c
c if ishape=1 print global x,y coords to file bshape
c if ishape=2 figure out symmetric coords as well
c
if(ishape.eq.1.or.ishape.eq.2.and.ncount.eq.1.and.isarch.eq.1)then
open(8,file='bshape',status='new')
do 1 ii=1,nnod
if(ishape.eq.2)then
xcoord(nnod-1+ii)=rad*cos(pi/2.0-coord(ii)/rad)
xcoord(nnod+1-ii)=-rad*cos(pi/2.0-coord(ii)/rad)
ycoord(nnod-1+ii)=rad*sin(pi/2.0-coord(ii)/rad)
ycoord(nnod+1-ii)=ycoord(nnod-1+ii)
else
xcoord(ii)=-rad*cos(pi/2.0+coord(ii)/rad-coord(nnod)/(2*rad))
    ycoord(ii)=rad*sin(pi/2.0+coord(ii)/rad-coord(nnod)/(2*rad))
endif
   1 continue
do 100 ii=1,nnod
 100 write(8,2000)xcoord(ii),ycoord(ii)
if(ishape.eq.2)then
do 110 ii=2,nnod
 110 write(8,2000)xcoord(nnod+ii-1),ycoord(nnod+ii-1)
endif
    write(8,2010)
endif
c
c global displacements for straight beams
c
if(ishape.eq.1.and.ncount.eq.1.and.isarch.eq.0)then
open(8,file='bshape',status='new')
do 2 ii=1,nnod
xcoord(ii)= coord(nnod)-coord(ii)
    ycoord(ii)=0.0
   2 write(8,2000)xcoord(ii),ycoord(ii)
    write(8,2010)
endif
c
c print out global displacements
c
write(6,1000)
write(6,1010)
write(6,1020)ncount,icount
write(6,1030)
```

```
do 90 ii=0,nnod-1
c
c x,y for arches
c
if(ishape.eq.1.and.isarch.eq.1)then
if(ishape.eq.1.and.isarch.eq.1.and.ii.eq.0)write(8,2020)ncount
xcoord(ii+1)=-(rad-gd(5*ii+3))*
     .     cos(pi/2.0+coord(ii+1)/rad-coord(nnod)/(2*rad)+
     . gd(5*ii+1)/rad)
ycoord(ii+1)=(rad-gd(5*ii+3))*
     .     sin(pi/2.0+coord(ii+1)/rad-coord(nnod)/(2*rad)+
     . gd(5*ii+1)/rad)
write(8,2000)xcoord(ii+1),ycoord(ii+1)
endif
if(ishape.eq.2.and.isarch.eq.1)then
if(ii.eq.0)write(8,2020)ncount
xcoord(nnod+ii)=(rad-gd(5*ii+3))*
     . cos(pi/2.0-coord(ii+1)/rad +gd(5*ii+1)/rad)
xcoord(nnod-ii)=-xcoord(nnod+ii)
ycoord(nnod+ii)=(rad-gd(5*ii+3))*
     . sin(pi/2.0-coord(ii+1)/rad+gd(5*ii+1)/rad)
ycoord(nnod-ii)=ycoord(nnod+ii)
endif
c
c x,y for straight beams
c
if(ishape.eq.1.and.isarch.eq.0)then
if(ishape.eq.1.and.isarch.eq.0.and.ii.eq.0)write(8,2020)ncount
xcoord(ii+1)=coord(nnod)-coord(ii+1)-gd(5*ii+1)
ycoord(ii+1)=-gd(5*ii+3)
write(8,2000)xcoord(ii+1),ycoord(ii+1)
endif
   write(6,1040)ii+1,(gd(5*ii+jj),jj=1,4)
  90 write(6,1050)gd(5*ii+5)
if(ishape.eq.2.and.isarch.eq.1)then
do 95 ii=1,2*nnod-1
  95 write(8,2000)xcoord(ii),ycoord(ii)
endif
if(ishape.ge.1)write(8,2010)
c
c compute equivalent forces requested
c
  3 do 5 ii=1,ndof
gf(ii)=0.0d0
do 5 jj=1,9
```

```
gstif(ii,jj)=0.0d0
  5 gn(ii,jj)=0.0d0
c
c loop over all elements for stiffness and forces
c
   9 do 30 ielem=1,nelem
do 10 ii=1,9
  10 eld(ii)=gd(ii+(ielem-1)*5)
c
call stiff(ielem,icount,ncount,kcall)
c
c Assemble global stiffness array, gstif, global equilibrium
c stiffness, gn, in banded form. Half-bandwidth=9.  Also
c assemble global force vector, gf.
c
nr=(ielem-1)*5 + 1
do 30 jj=0,8
gf(nr+jj)=gf(nr+jj)+elp(jj+1)
do 30 kk=1,9-jj
gstif(nr+jj,kk)=gstif(nr+jj,kk)+stif(jj+1,kk+jj)
if(linear.eq.1)goto 30
   gn(nr+jj,kk)=gn(nr+jj,kk)+eln(jj+1,kk+jj)
  30 continue
c
c calculate the residual force vector for nonlinear
c analysis. -[gn]*{gd}+R=-[k+n1/2+n2/3]*{q}+R=gf
c
       do 60 jj=1,nforc
ii=iforc(jj)
       add=0.
       do 50 kk=1,ii-1
       if(ii-kk+1 .gt. 9)goto 50
       add=add+gn(kk,ii-kk+1)*gd(kk)
   50  continue
       res=0.
       do 55 ll=1,9
       if(ll+ii-1 .gt. ndof)goto 55
       res=res + gn(ii,ll)*gd(ll+ii-1)
   55  continue
c
c compute nodal force
c
vforc(jj)=res+add
   60 continue
c
```

```
c print nodal forces and create plot file
c
open(7,file='plot',status='new')
if(ncount.eq.1)write(7,*)0.0,0.0
write(6,1060)
do 70 ii=1,nforc
write(7,1065)gd(iforc(ii)),gd(iforc(ii)-2),vforc(ii)
   70 write(6,1070)iforc(ii),vforc(ii)
 1000 format(/)
 1010 format(1x,'Results of nonlinear analysis')
 1020 format(1x,'increment=',i3,'   iteration=',i3)
 1030 format(1x,'Node',7x,'V',13x,'Psi-s',13x,'W',13x,'W-s')
 1040 format(1x,i4,4(2x,d12.5))
 1050 format(1x,'Midnode v:',3x,d12.5)
 1060 format(1x,/,'Equivalent nodal forces:')
 1065 format(1x,f12.5,2x,f12.5,2x,f12.5)
 1070 format(1x,'DOF no:',i4,2x,'Force:',1x,d12.5)
 2000 format(1x,f12.5,2x,f12.5)
 2010  format(//)
 2020 format(/,1x,i4)
return
end
c
c
c
subroutine beamk(bmk,ek1,p1)
c
implicit double precision (a-h,o-z)
c
common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
dimension bmk(7,7)
do 10 ii=1,7
do 10 jj=1,7
  10 bmk(jj,ii)=0.0d0
c
       bmk(2,2)=fe*p1**4-(2*de*p1**2)+ae
c
       bmk(2,3)=de*p1**3-(ae*p1)
c
       bmk(2,5)=-(he*ek1*p1**3)+fe*ek1*p1
c
       bmk(2,7)=-(he*ek1*p1**3)+fe*(-p1**3+ek1*p1)+de*p1
c
       bmk(3,3)=de*p1**4+ae*p1**2
c
```

```
      bmk(3,5)=-(2*fe*ek1*p1**2)
c
      bmk(3,7)=-(2*fe*ek1*p1**2)-(2*de*p1**2)
c
      bmk(5,5)=ej*ek1**2*p1**2+he*ek1**2
c
      bmk(5,7)=he*(ek1*p1**2+ek1**2)+ej*ek1**2*p1**2+fe*ek1
c
      bmk(7,7)=he*(2*ek1*p1**2+ek1**2)+fe*(p1**2+2*ek1)+ej*ek1**
     . 2*p1**2+de
c
      bmk(4,4)=9*fs*ek1**2+6*ds*ek1+as
c
      bmk(4,6)=9*fs*ek1**2+6*ds*ek1+as
c
      bmk(6,6)=9*fs*ek1**2+6*ds*ek1+as
c
do 100 ii=1,7
do 100 jj=ii,7
 100 bmk(jj,ii)=bmk(ii,jj)
return
end
c
c
c
       subroutine beamn1(q,bmn1,ek,p1)
c
c Note that k1 appears as 'ek' in this subroutine
c
c The equations in this subroutine were generated by MACSYMA.
c
implicit double precision (a-h,o-z)
c
common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
dimension bmn1(7,7),q(7)
c
      bmn1(1,1)=-(he*(q(7)+q(5))*ek*p1**5)-(fe*p1**3*(2*q(3)*p1
     . **4-(3*q(2)*p1**3)+q(7)*p1**2-(q(7)*ek)-(q(5)*ek)))+de*p1**3
     . *(p1*(3*q(3)*p1-(4*q(2)))+q(7))+ae*p1**2*(-(q(3)*p1)+q(2))
c
c
      bmn1(1,2)=-(he*(q(6)+q(4))*ek*p1**5)+fe*p1**3*(3*q(1)*p1**
     . 3-(q(6)*p1**2)+2*q(4)*p1**2+q(6)*ek+q(4)*ek)-(de*p1**3*(4*q(
     . 1)*p1-q(6)+3*q(4)))+ae*p1*(q(1)*p1+q(4))
c
```

```
c
      bmn1(1,3)=2*he*(q(6)+q(4))*ek*p1**6-(2*fe*p1**4*(q(1)*p1**
     . 3-(q(6)*p1**2)+q(6)*ek+q(4)*ek))+de*p1**4*(3*q(1)*p1-(2*q(6)
     . )+q(4))-(ae*p1**2*(q(1)*p1+q(4)))
c
c
      bmn1(1,4)=he*ek*p1**2*(2*q(3)*p1**4-(q(2)*p1**3)-(3*q(7)*p1**2
     . )-(2*q(5)*p1**2)+q(7)*ek+q(5)*ek)-(ej*(q(7)+q(5))*ek**2*p1**4
     . )+fe*p1**2*(2*q(2)*p1**3-(2*q(3)*ek*p1**2)-(2*q(7)*p1**2)+q(2
     . )*ek*p1+3*q(7)*ek+2*q(5)*ek)+de*p1**2*(p1*(q(3)*p1-(3*q(2)))
     . +2*q(7))+ae*p1*(-(q(3)*p1)+q(2))
c
c
      bmn1(1,5)=-(ej*(q(6)+q(4))*ek**2*p1**4)-(he*ek*p1**2*(q(1)
     . *p1**3+q(6)*p1**2+2*q(4)*p1**2-(q(6)*ek)-(q(4)*ek)))+fe*ek*p1
     . **2*(q(1)*p1+q(6)+2*q(4))
c
c
      bmn1(1,6)=fe*p1**2*(2*q(3)*p1**4-(q(2)*p1**3)-(2*q(3)*ek*p1**2
     . )-(q(7)*p1**2)+q(2)*ek*p1+2*q(7)*ek+q(5)*ek)+he*ek*p1**2*(2*q
     . (3)*p1**4-(q(2)*p1**3)-(2*q(7)*p1**2)-(q(5)*p1**2)+q(7)*ek+q(5)*
     . ek)-(ej*(q(7)+q(5))*ek**2*p1**4)-(de*p1**2*(p1*(2*q(3)*
     . p1-q(2))-q(7)))
c
c
      bmn1(1,7)=-(ej*(q(6)+q(4))*ek**2*p1**4)-(he*ek*p1**2*(q(1)
     . *p1**3+2*q(6)*p1**2+3*q(4)*p1**2-(q(6)*ek)-(q(4)*ek)))-(fe*p1
     . **2*(q(1)*p1**3+q(6)*p1**2+2*q(4)*p1**2-(q(1)*ek*p1)-(2*q(6)*ek)-
     . (3*q(4)*ek)))+de*p1**2*(q(1)*p1+q(6)+2*q(4))
c
c
      bmn1(2,2)=-(3*fe*p1*(2*q(3)*p1**4-(3*q(2)*p1**3)+q(7)*p1**2-(q
     . (7)*ek)-(q(5)*ek)))-(3*he*(q(7)+q(5))*ek*p1**3)+3*de*p1*
     . (p1*(3*q(3)*p1-(4*q(2)))+q(7))-(3*ae*(q(3)*p1-q(2)))
c
c
      bmn1(2,3)=6*he*(q(7)+q(5))*ek*p1**4-(6*fe*p1**2*(q(2)*p1**
     . 3-(q(7)*p1**2)+q(7)*ek+q(5)*ek))-(3*de*p1**2*(p1*(q(3)*p1-(3
     . *q(2)))+2*q(7)))+3*ae*p1*(q(3)*p1-q(2))
c
c
      bmn1(2,4)=-(ej*(q(6)+q(4))*ek**2*p1**4)+fe*p1**2*(2*q(1)*
     . p1**3-(2*q(6)*p1**2)+q(1)*ek*p1+3*q(6)*ek+4*q(4)*ek)-(he*ek*
     . p1**2*(q(1)*p1**3+3*q(6)*p1**2+4*q(4)*p1**2-(q(6)*ek)-(q(4)*ek)))
     . -(de*p1**2*(3*q(1)*p1-(2*q(6))+q(4)))+ae*(q(1)*p1+q(4))
```

```
c
c
      bmn1(2,5)=3*he*ek*(2*q(3)*p1**4-(q(2)*p1**3)-(q(7)*p1**2)+q(7)
     . *ek+q(5)*ek)-(3*ej*(q(7)+q(5))*ek**2*p1**2)-(3*fe*ek*(p1*
     . (2*q(3)*p1-q(2))-q(7)))
c
c
      bmn1(2,6)=-(ej*(q(6)+q(4))*ek**2*p1**4)-(he*ek*p1**2*(q(1)
     . *p1**3+2*q(6)*p1**2+3*q(4)*p1**2-(q(6)*ek)-(q(4)*ek)))-(fe*p1
     . **2*(q(1)*p1**3+q(6)*p1**2+2*q(4)*p1**2-(q(1)*ek*p1)-(2*q(6)*ek)-
     . (3*q(4)*ek)))+de*p1**2*(q(1)*p1+q(6)+2*q(4))
c
c
      bmn1(2,7)=3*fe*(2*q(3)*p1**4-(q(2)*p1**3)-(2*q(3)*ek*p1**2)-(q
     . (7)*p1**2)+q(2)*ek*p1+2*q(7)*ek+q(5)*ek)+3*he*ek*(2*q(3)*p1**
     . 4-(q(2)*p1**3)-(2*q(7)*p1**2)-(q(5)*p1**2)+q(7)*ek+q(5)*ek)-(3*ej
     . *(q(7)+q(5))*ek**2*p1**2)-(3*de*(p1*(2*q(3)*p1-q(2))-q(7
     . )))
c
c
      bmn1(3,3)=9*fe*(q(7)+q(5))*ek*p1**3-(3*de*p1**3*(p1*(2*q(
     . 3)*p1+q(2))-(3*q(7))))-(3*ae*p1**2*(q(3)*p1-q(2)))
c
c
      bmn1(3,4)=-(2*ej*(q(6)+q(4))*ek**2*p1**5)+he*ek*p1**3*(2*q
     . (1)*p1**3-(2*q(6)*p1**2)-(q(6)*ek)-(q(4)*ek))-(2*fe*ek*p1**3*
     . (q(1)*p1+2*q(6)+3*q(4)))+de*p1**3*(q(1)*p1-(3*q(6))-(2*q(4))
     . )+ae*p1*(-(q(1)*p1)-q(4))
c
c
      bmn1(3,5)=3*he*ek*p1*(2*q(2)*p1**3-(2*q(7)*p1**2)-(q(7)*ek)-(q
     . (5)*ek))-(6*ej*(q(7)+q(5))*ek**2*p1**3)+3*fe*ek*p1*(p1*(3
     . *q(3)*p1-(2*q(2)))-q(7))
c
c
      bmn1(3,6)=-(2*ej*(q(6)+q(4))*ek**2*p1**5)+he*ek*p1**3*(2*q
     . (1)*p1**3-(4*q(6)*p1**2)-(2*q(4)*p1**2)-(q(6)*ek)-(q(4)*ek))+2*fe
     . *p1**3*(q(1)*p1**3-(q(6)*p1**2)-(q(1)*ek*p1)-(q(6)*ek)-(2*q(4
     . )*ek))-(de*p1**3*(2*q(1)*p1+q(6)+3*q(4)))
c
c
      bmn1(3,7)=3*fe*p1*(2*q(2)*p1**3+3*q(3)*ek*p1**2-(2*q(7)*p1**2)
     . -(2*q(2)*ek*p1)-(2*q(7)*ek)-(q(5)*ek))+3*he*ek*p1*(2*q(2)*p1
     . **3-(4*q(7)*p1**2)-(2*q(5)*p1**2)-(q(7)*ek)-(q(5)*ek))-(6*ej*
     . (q(7)+q(5))*ek**2*p1**3)+3*de*p1*(p1*(3*q(3)*p1-(2*q(2)))-q(
     . 7))
c
c
      bmn1(4,4)=-(he*ek*p1*(4*q(2)*p1**3+q(3)*ek*p1**2-(4*q(7)*p1**2
     . )-(q(2)*ek*p1)-(2*q(7)*ek)-(2*q(5)*ek)))+3*el*(q(7)+q(5))*ek
     . **3*p1**3-(fe*ek*p1*(2*p1*(3*q(3)*p1-(2*q(2)))-(5*q(7))-(3*q(
     . 5))))-(de*p1*(p1*(2*q(3)*p1+q(2))-(3*q(7))))-(ej*ek**2*
     . p1**3*(p1*(2*q(3)*p1+q(2))-(7*q(7))-(4*q(5))))+ae*(-(q(3)*p1)
     . +q(2))
c
c
      bmn1(4,5)=-(he*ek*p1*(2*q(1)*p1**3-(2*q(6)*p1**2)-(q(1)*ek*p1)
     . -(q(6)*ek)-(2*q(4)*ek)))+3*el*(q(6)+q(4))*ek**3*p1**3+fe*
     . ek*p1*(2*q(1)*p1+q(6)+3*q(4))-(ej*ek**2*p1**3*(q(1)*p1-(5*q(6
     . ))-(4*q(4))))
c
c
      bmn1(4,6)=-(he*ek*p1*(2*q(3)*p1**4+3*q(2)*p1**3+q(3)*ek*p1**2-
     . (7*q(7)*p1**2)-(2*q(5)*p1**2)-(q(2)*ek*p1)-(q(7)*ek)-(q(5)*ek))-
     . (fe*p1*(2*q(2)*p1**3+4*q(3)*ek*p1**2-(2*q(7)*p1**2)-(3*q(2)*
     . ek*p1)-(2*q(7)*ek)-(q(5)*ek)))+3*el*(q(7)+q(5))*ek**3*p1**3-(
     . de*p1*(p1*(3*q(3)*p1-(2*q(2)))-q(7)))-(ej*ek**2*p1**3*(
     . p1*(2*q(3)*p1+q(2))-(8*q(7))-(5*q(5))))
c
c
      bmn1(4,7)=-(he*ek*p1*(3*q(1)*p1**3-(7*q(6)*p1**2)-(4*q(4)*p1**
     . 2)-(q(1)*ek*p1)-(q(6)*ek)-(2*q(4)*ek)))-(fe*p1*(2*q(1)*p1**3-
     . (2*q(6)*p1**2)-(3*q(1)*ek*p1)-(2*q(6)*ek)-(5*q(4)*ek)))+3*el*
     . (q(6)+q(4))*ek**3*p1**3+de*p1*(2*q(1)*p1+q(6)+3*q(4))-(ej
     . *ek**2*p1**3*(q(1)*p1-(8*q(6))-(7*q(4))))
c
c
      bmn1(5,5)=-(3*ej*ek**2*p1*(p1*(2*q(3)*p1+q(2))-(3*q(7))))-(3*he
     . *ek**2*(q(3)*p1-q(2)))+9*el*(q(7)+q(5))*ek**3*p1
c
c
      bmn1(5,6)=-(he*ek*p1*(q(1)*p1**3-(3*q(6)*p1**2)-(2*q(4)*p1**2)
     . -(q(1)*ek*p1)-(q(4)*ek)))+3*el*(q(6)+q(4))*ek**3*p1**3-(ej
     . *ek**2*p1**3*(q(1)*p1-(6*q(6))-(5*q(4))))+fe*ek*p1*(q(1)*p1+
     . q(4))
c
c
      bmn1(5,7)=-(3*he*ek*(p1*(p1*(2*q(3)*p1+q(2))+q(3)*ek-(3*q(7)))
     . -(q(2)*ek)))-(3*ej*ek**2*p1*(p1*(2*q(3)*p1+q(2))-(6*q(7))-(3*
     . q(5))))-(3*fe*ek*(q(3)*p1-q(2)))+9*el*(q(7)+q(5))*ek**3*
     . p1
c
c
      bmn1(6,6)=3*el*(q(7)+q(5))*ek**3*p1**3-(he*ek*p1**2*(p1*(2
     . *p1*(2*q(3)*p1+q(2))+q(3)*ek-(9*q(7))-(3*q(5)))-(q(2)*ek)))-(fe
     . *p1**2*(p1*(p1*(2*q(3)*p1+q(2))+2*q(3)*ek-(3*q(7)))-(2*q(2)*ek)
     . ))-(ej*ek**2*p1**3*(p1*(2*q(3)*p1+q(2))-(9*q(7))-(6*q(5))))+
     . de*p1**2*(-(q(3)*p1)+q(2))
c
c
      bmn1(6,7)=-(he*ek*p1*(2*q(1)*p1**3-(9*q(6)*p1**2)-(7*q(4)*p1**
     . 2)-(q(1)*ek*p1)-(q(4)*ek)))-(fe*p1*(q(1)*p1**3-(3*q(6)*p1**2)
     . -(2*q(4)*p1**2)-(2*q(1)*ek*p1)-(2*q(4)*ek)))+3*el*(q(6)+q(4))
     . *ek**3*p1**3-(ej*ek**2*p1**3*(q(1)*p1-(9*q(6))-(8*q(4))))+de
     . *p1*(q(1)*p1+q(4))
c
c
      bmn1(7,7)=-(3*he*ek*(p1*(2*p1*(2*q(3)*p1+q(2))+q(3)*ek-(9*q(7)
     . )-(3*q(5)))-(q(2)*ek)))-(3*fe*(p1*(p1*(2*q(3)*p1+q(2))+2*q(3)
     . *ek-(3*q(7)))-(2*q(2)*ek)))-(3*ej*ek**2*p1*(p1*(2*q(3)*p1+q(2
     . ))-(9*q(7))-(6*q(5))))-(3*de*(q(3)*p1-q(2)))+9*el*(q(7)+
     . q(5))*ek**3*p1
c
do 100 ii=1,7
do 100 jj=ii,7
 100 bmn1(jj,ii)=bmn1(ii,jj)
return
end
c
c
c
subroutine beamn2(q,bmn2,ek,p1)
c
c Note that k1 appears as 'ek' in this subroutine
c
c The equations in this subroutine were generated by MACSYMA.
c
implicit double precision (a-h,o-z)
c
common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
dimension bmn2(7,7),q(7)
c
      t0=he*p1**2*(36*q(1)**2*p1**8-(72*q(1)*q(6)*p1**7)+36*q(6)**2*
     . p1**6+12*q(2)**2*p1**6-(24*q(2)*q(7)*p1**5)+12*q(3)*q(7)*ek*p1**4
     . -(44*q(6)**2*ek*p1**4)-(88*q(4)*q(6)*ek*p1**4)+12*q(3)*q(5)*ek*p1
     . **4-(44*q(4)**2*ek*p1**4)+12*q(7)**2*p1**4+7*q(6)**2*ek**2*p1**2+
     . 14*q(4)*q(6)*ek**2*p1**2+7*q(4)**2*ek**2*p1**2-(12*q(7)**2*ek*p1
     . **2)-(12*q(5)*q(7)*ek*p1**2)+q(7)**2*ek**2+2*q(5)*q(7)*ek**2+q(5)
     . **2*ek**2)/6.0
       t0=t0+fe*p1**2*(8*q(3)**2*p1**6+81*q(1)**2*p1**6-(30*q(1)*q(6)
     . *p1**5)+132*q(1)*q(4)*p1**5-(44*q(2)*q(3)*p1**5)+28*q(3)*q(7)*p1
     . **4-(19*q(6)**2*p1**4)-(68*q(4)*q(6)*p1**4)+32*q(4)**2*p1**4+27*q
     . (2)**2*p1**4+6*q(1)*q(6)*ek*p1**3+6*q(1)*q(4)*ek*p1**3-(10*q(2)*q
     . (7)*p1**3)-(4*q(3)*q(7)*ek*p1**2)+10*q(6)**2*ek*p1**2+26*q(4)*q(6
     . )*ek*p1**2-(4*q(3)*q(5)*ek*p1**2)+16*q(4)**2*ek*p1**2-(9*q(7)**2*
     . p1**2)+2*q(2)*q(7)*ek*p1+2*q(2)*q(5)*ek*p1+2*q(7)**2*ek+2*q(5)*q(
     . 7)*ek)/6.0
       bmn2(1,1)=t0-(1.0/3.0*ej*ek*p1**4*(24*q(1)*q(6)*p1**5+24*q(1)*
     . q(4)*p1**5-(24*q(6)**2*p1**4)-(24*q(4)*q(6)*p1**4)+8*q(2)*q(7)*p1
     . **3+8*q(2)*q(5)*p1**3+8*q(6)**2*ek*p1**2+16*q(4)*q(6)*ek*p1**2+8*
     . q(4)**2*ek*p1**2-(8*q(7)**2*p1**2)-(8*q(5)*q(7)*p1**2)+3*q(7)**2*
     . ek+6*q(5)*q(7)*ek+3*q(5)**2*ek))-(1.0/6.0*de*p1**2*(3*q(3)**
     . 2*p1**4+18*q(1)**2*p1**4-(6*q(1)*q(6)*p1**3)+30*q(1)*q(4)*p1**3-(
     . 10*q(2)*q(3)*p1**3)+4*q(3)*q(7)*p1**2-(7*q(6)**2*p1**2)-(20*q(4)*
     . q(6)*p1**2)+5*q(4)**2*p1**2+6*q(2)**2*p1**2-(2*q(2)*q(7)*p1)-q(7)
     . **2))+4.0/3.0*el*ek**2*p1**6*(4*(q(6)+q(4))**2*p1**2+(q(7)+q(
     . 5))**2)+ae*p1**2*(q(3)**2*p1**2+3*q(1)**2*p1**2-(q(2)*(2*q(3)
     . *p1-q(2)))+6*q(1)*q(4)*p1+3*q(4)**2)/2.0
c
c
       t0=he*p1**2*(12*q(1)*q(2)*p1**6-(12*q(1)*q(7)*p1**5)-(12*q(2)*
     . q(6)*p1**5)+8*q(3)*q(6)*ek*p1**4+8*q(3)*q(4)*ek*p1**4+12*q(6)*q(7
     . )*p1**4-(16*q(6)*q(7)*ek*p1**2)-(16*q(4)*q(7)*ek*p1**2)-(8*q(5)*q
     . (6)*ek*p1**2)-(8*q(4)*q(5)*ek*p1**2)+3*q(6)*q(7)*ek**2+3*q(4)*q(7
     . )*ek**2+3*q(5)*q(6)*ek**2+3*q(4)*q(5)*ek**2)/3.0+4*el*(q(6)+q
     . (4))*(q(7)+q(5))*ek**2*p1**6-(1.0/3.0*fe*p1**2*(22*q(1)*q(3)*
     . p1**5-(10*q(3)*q(6)*p1**4)+12*q(3)*q(4)*p1**4-(27*q(1)*q(2)*p1**4
     . )+5*q(1)*q(7)*p1**3+5*q(2)*q(6)*p1**3-(22*q(2)*q(4)*p1**3)+3*q(3)
     . *q(6)*ek*p1**2+3*q(3)*q(4)*ek*p1**2+5*q(6)*q(7)*p1**2+10*q(4)*q(7
     . )*p1**2-(q(1)*q(7)*ek*p1)-(q(2)*q(6)*ek*p1)-(q(1)*q(5)*ek*p1)-(q(
     . 2)*q(4)*ek*p1)-(4*q(6)*q(7)*ek)-(5*q(4)*q(7)*ek)-(2*q(5)*q(6)*ek)
     . -(3*q(4)*q(5)*ek)))
       bmn2(1,2)=t0-(1.0/3.0*ej*ek*p1**4*(8*q(1)*q(7)*p1**3+8*q(2)*q(
     . 6)*p1**3+8*q(1)*q(5)*p1**3+8*q(2)*q(4)*p1**3-(16*q(6)*q(7)*p1**2)
     . -(8*q(4)*q(7)*p1**2)-(8*q(5)*q(6)*p1**2)+5*q(6)*q(7)*ek+5*q(4)*q(
     . 7)*ek+5*q(5)*q(6)*ek+5*q(4)*q(5)*ek))+de*p1**2*(5*q(1)*q(3)*
     . p1**3-(4*q(3)*q(6)*p1**2)+q(3)*q(4)*p1**2-(6*q(1)*q(2)*p1**2)+q(1
     . )*q(7)*p1+q(2)*q(6)*p1-(5*q(2)*q(4)*p1)+3*q(6)*q(7)+4*q(4)*q(7))/
     . 3.0-(ae*p1*(q(1)*p1+q(4))*(q(3)*p1-q(2)))
c
```

```
c
      t0=fe*p1**3*(8*q(1)*q(3)*p1**5-(8*q(3)*q(6)*p1**4)-(22*q(1)*q(
     . 2)*p1**4)+14*q(1)*q(7)*p1**3+10*q(2)*q(6)*p1**3-(12*q(2)*q(4)*p1
     . **3)+5*q(3)*q(6)*ek*p1**2+5*q(3)*q(4)*ek*p1**2-(2*q(6)*q(7)*p1**2
     . )+12*q(4)*q(7)*p1**2-(2*q(1)*q(7)*ek*p1)-(3*q(2)*q(6)*ek*p1)-(2*q
     . (1)*q(5)*ek*p1)-(3*q(2)*q(4)*ek*p1)-(4*q(6)*q(7)*ek)-(6*q(4)*q(7)
     . *ek)-(2*q(5)*q(6)*ek)-(4*q(4)*q(5)*ek))/3.0-(2.0/3.0*ej*(q(6)
     . +q(4))*(q(7)+q(5))*ek**2*p1**5)-(1.0/3.0*he*ek*p1**3*(8*q(3)*
     . q(6)*p1**4+8*q(3)*q(4)*p1**4-(6*q(1)*q(7)*p1**3)-(8*q(2)*q(6)*p1
     . **3)-(6*q(1)*q(5)*p1**3)-(8*q(2)*q(4)*p1**3)-(2*q(6)*q(7)*p1**2)-
     . (8*q(4)*q(7)*p1**2)-(2*q(5)*q(6)*p1**2)-(8*q(4)*q(5)*p1**2)+3*q(6
     . )*q(7)*ek+3*q(4)*q(7)*ek+3*q(5)*q(6)*ek+3*q(4)*q(5)*ek))
      bmn2(1,3)=t0-(1.0/3.0*de*p1**3*(3*q(1)*q(3)*p1**3-(7*q(3)*q(6
     . )*p1**2)-(4*q(3)*q(4)*p1**2)-(5*q(1)*q(2)*p1**2)+2*q(1)*q(7)*p1+4
     . *q(2)*q(6)*p1-(q(2)*q(4)*p1)+3*q(6)*q(7)+5*q(4)*q(7)))+ae*p1
     . **2*(q(1)*p1+q(4))*(q(3)*p1-q(2))
c
c
      t0=-(1.0/6.0*he*ek*p1*(8*q(3)**2*p1**6+88*q(1)*q(6)*p1**5+88*q
     . (1)*q(4)*p1**5-(16*q(2)*q(3)*p1**5)-(16*q(3)*q(7)*p1**4)-(16*q(6)
     . **2*p1**4)+56*q(4)*q(6)*p1**4-(16*q(3)*q(5)*p1**4)+72*q(4)**2*p1
     . **4-(14*q(1)*q(6)*ek*p1**3)-(14*q(1)*q(4)*ek*p1**3)+32*q(2)*q(7)*
     . p1**3+16*q(2)*q(5)*p1**3+6*q(3)*q(7)*ek*p1**2-(7*q(6)**2*ek*p1**2
     . )-(28*q(4)*q(6)*ek*p1**2)+6*q(3)*q(5)*ek*p1**2-(21*q(4)**2*ek*p1
     . **2)-(8*q(7)**2*p1**2)-(6*q(2)*q(7)*ek*p1)-(6*q(2)*q(5)*ek*p1)-(q
     . (7)**2*ek)-(2*q(5)*q(7)*ek)-(q(5)**2*ek)))
      t0=t0+fe*p1*(66*q(1)**2*p1**6-(68*q(1)*q(6)*p1**5)+64*q(1)*q(4
     . )*p1**5-(24*q(2)*q(3)*p1**5)+5*q(3)**2*ek*p1**4+3*q(1)**2*ek*p1**
     . 4+24*q(3)*q(7)*p1**4+2*q(6)**2*p1**4-(64*q(4)*q(6)*p1**4)+22*q(2)
     . **2*p1**4+26*q(1)*q(6)*ek*p1**3+32*q(1)*q(4)*ek*p1**3-(6*q(2)*q(3
     . )*ek*p1**3)-(20*q(2)*q(7)*p1**3)-(12*q(3)*q(7)*ek*p1**2)+10*q(6)
     . **2*ek*p1**2+46*q(4)*q(6)*ek*p1**2-(8*q(3)*q(5)*ek*p1**2)+39*q(4)
     . **2*ek*p1**2+q(2)**2*ek*p1**2-(2*q(7)**2*p1**2)+10*q(2)*q(7)*ek*
     . p1+6*q(2)*q(5)*ek*p1+2*q(7)**2*ek+2*q(5)*q(7)*ek)/6.0
      t0=t0-(1.0/3.0*ej*ek*p1**3*(12*q(1)**2*p1**6-(24*q(1)*q(6)*p1
     . **5)+12*q(6)**2*p1**4+4*q(2)**2*p1**4+16*q(1)*q(6)*ek*p1**3+16*q(
     . 1)*q(4)*ek*p1**3-(8*q(2)*q(7)*p1**3)+2*q(3)*q(7)*ek*p1**2-(22*q(6
     . )**2*ek*p1**2)-(28*q(4)*q(6)*ek*p1**2)+2*q(3)*q(5)*ek*p1**2-(6*q(
     . 4)**2*ek*p1**2)+4*q(7)**2*p1**2+5*q(2)*q(7)*ek*p1+5*q(2)*q(5)*ek*
     . p1-(7*q(7)**2*ek)-(7*q(5)*q(7)*ek)))+el*ek**2*p1**3*(32*q(1)*
     . q(6)*p1**5+32*q(1)*q(4)*p1**5-(32*q(6)**2*p1**4)-(32*q(4)*q(6)*p1
     . **4)+12*q(2)*q(7)*p1**3+12*q(2)*q(5)*p1**3+9*q(6)**2*ek*p1**2+18*
     . q(4)*q(6)*ek*p1**2+9*q(4)**2*ek*p1**2-(12*q(7)**2*p1**2)-(12*q(5)
     . *q(7)*p1**2)+3*q(7)**2*ek+6*q(5)*q(7)*ek+3*q(5)**2*ek)/3.0
      bmn2(1,4)=t0+de*p1*(4*q(3)**2*p1**4-(15*q(1)**2*p1**4)+20*q(1
     . )*q(6)*p1**3-(10*q(1)*q(4)*p1**3)+2*q(2)*q(3)*p1**3-(10*q(3)*q(7)
     . *p1**2)+7*q(6)**2*p1**2+34*q(4)*q(6)*p1**2+12*q(4)**2*p1**2-(5*q(
     . 2)**2*p1**2)+8*q(2)*q(7)*p1+q(7)**2)/6.0-(4.0/3.0*re*ek**3*p1
     . **5*(3*(q(6)+q(4))**2*p1**2+(q(7)+q(5))**2))+ae*p1*(q(3)**2*
     . p1**2+3*q(1)**2*p1**2-(q(2)*(2*q(3)*p1-q(2)))+6*q(1)*q(4)*p1+3*q(
     . 4)**2)/2.0
c
c
      t0=he*ek*p1*(6*q(1)*q(3)*p1**5+2*q(3)*q(6)*p1**4+8*q(3)*q(4)*
     . p1**4-(6*q(1)*q(7)*p1**3)-(8*q(2)*q(6)*p1**3)-(8*q(2)*q(4)*p1**3)
     . -(3*q(3)*q(6)*ek*p1**2)-(3*q(3)*q(4)*ek*p1**2)+6*q(6)*q(7)*p1**2+
     . q(1)*q(7)*ek*p1+3*q(2)*q(6)*ek*p1+q(1)*q(5)*ek*p1+3*q(2)*q(4)*ek*
     . p1+q(4)*q(7)*ek+q(4)*q(5)*ek)/3.0-(8.0/3.0*re*(q(6)+q(4))*(q(
     . 7)+q(5))*ek**3*p1**5)-(1.0/3.0*ej*ek*p1**3*(8*q(1)*q(2)*p1**4
     . -(8*q(1)*q(7)*p1**3)-(8*q(2)*q(6)*p1**3)+2*q(3)*q(6)*ek*p1**2+2*q
     . (3)*q(4)*ek*p1**2+8*q(6)*q(7)*p1**2+6*q(1)*q(7)*ek*p1+5*q(2)*q(6)
     . *ek*p1+6*q(1)*q(5)*ek*p1+5*q(2)*q(4)*ek*p1-(13*q(6)*q(7)*ek)-(7*q
     . (4)*q(7)*ek)-(6*q(5)*q(6)*ek)))
      bmn2(1,5)=t0+2.0/3.0*el*ek**2*p1**3*(4*q(1)*q(7)*p1**3+6*q(2)*
     . q(6)*p1**3+4*q(1)*q(5)*p1**3+6*q(2)*q(4)*p1**3-(10*q(6)*q(7)*p1**
     . 2)-(6*q(4)*q(7)*p1**2)-(4*q(5)*q(6)*p1**2)+3*q(6)*q(7)*ek+3*q(4)*
     . q(7)*ek+3*q(5)*q(6)*ek+3*q(4)*q(5)*ek)-(1.0/3.0*fe*ek*p1*(2*q
     . (1)*q(3)*p1**3+2*q(3)*q(6)*p1**2+4*q(3)*q(4)*p1**2-(q(1)*q(2)*p1
     . **2)-(q(1)*q(7)*p1)-(2*q(2)*q(6)*p1)-(3*q(2)*q(4)*p1)-(q(4)*q(7))
     . ))
c
c
      t0=-(1.0/3.0*he*p1**2*(18*q(1)**2*p1**7-(36*q(1)*q(6)*p1**6)+4
     . *q(3)**2*ek*p1**5+18*q(6)**2*p1**5+6*q(2)**2*p1**5+44*q(1)*q(6)*
     . ek*p1**4+44*q(1)*q(4)*ek*p1**4-(8*q(2)*q(3)*ek*p1**4)-(12*q(2)*q(
     . 7)*p1**4)-(2*q(3)*q(7)*ek*p1**3)-(30*q(6)**2*ek*p1**3)-(16*q(4)*q
     . (6)*ek*p1**3)-(2*q(3)*q(5)*ek*p1**3)+14*q(4)**2*ek*p1**3+6*q(7)**
     . 2*p1**3-(7*q(1)*q(6)*ek**2*p1**2)-(7*q(1)*q(4)*ek**2*p1**2)+16*q(
     . 2)*q(7)*ek*p1**2+8*q(2)*q(5)*ek*p1**2+3*q(3)*q(7)*ek**2*p1-(7*q(4
     . )*q(6)*ek**2*p1)+3*q(3)*q(5)*ek**2*p1-(7*q(4)**2*ek**2*p1)-(10*q(
     . 7)**2*ek*p1)-(6*q(5)*q(7)*ek*p1)-(3*q(2)*q(7)*ek**2)-(3*q(2)*q(5)
     . *ek**2)))
      t0=t0-(1.0/3.0*ej*ek*p1**3*(12*q(1)**2*p1**6-(48*q(1)*q(6)*p1
     . **5)-(24*q(1)*q(4)*p1**5)+36*q(6)**2*p1**4+24*q(4)*q(6)*p1**4+4*q
     . (2)**2*p1**4+16*q(1)*q(6)*ek*p1**3+16*q(1)*q(4)*ek*p1**3-(16*q(2)
     . *q(7)*p1**3)-(8*q(2)*q(5)*p1**3)+2*q(3)*q(7)*ek*p1**2-(30*q(6)**2
     . *ek*p1**2)-(44*q(4)*q(6)*ek*p1**2)+2*q(3)*q(5)*ek*p1**2-(14*q(4)
     . **2*ek*p1**2)+12*q(7)**2*p1**2+8*q(5)*q(7)*p1**2+5*q(2)*q(7)*ek*
     . p1+5*q(2)*q(5)*ek*p1-(10*q(7)**2*ek)-(13*q(5)*q(7)*ek)-(3*q(5)**2
     . *ek)))
```

```
      t0=t0+el*ek**2*p1**3*(32*q(1)*q(6)*p1**5+32*q(1)*q(4)*p1**5-(
     . 48*q(6)**2*p1**4)-(64*q(4)*q(6)*p1**4)-(16*q(4)**2*p1**4)+12*q(2)
     . *q(7)*p1**3+12*q(2)*q(5)*p1**3+9*q(6)**2*ek*p1**2+18*q(4)*q(6)*ek
     . *p1**2+9*q(4)**2*ek*p1**2-(16*q(7)**2*p1**2)-(20*q(5)*q(7)*p1**2)
     . -(4*q(5)**2*p1**2)+3*q(7)**2*ek+6*q(5)*q(7)*ek+3*q(5)**2*ek)/3.0
      bmn2(1,6)=t0-(1.0/6.0*fe*p1**2*(8*q(3)**2*p1**5+15*q(1)**2*p1
     . **5+38*q(1)*q(6)*p1**4+68*q(1)*q(4)*p1**4-(20*q(2)*q(3)*p1**4)-(5
     . *q(3)**2*ek*p1**3)-(3*q(1)**2*ek*p1**3)+4*q(3)*q(7)*p1**3-(21*q(6
     . )**2*p1**3)-(4*q(4)*q(6)*p1**3)+32*q(4)**2*p1**3+5*q(2)**2*p1**3-
     . (20*q(1)*q(6)*ek*p1**2)-(26*q(1)*q(4)*ek*p1**2)+6*q(2)*q(3)*ek*p1
     . **2+10*q(2)*q(7)*p1**2+8*q(3)*q(7)*ek*p1-(20*q(4)*q(6)*ek*p1)+4*q
     . (3)*q(5)*ek*p1-(23*q(4)**2*ek*p1)-(q(2)**2*ek*p1)-(7*q(7)**2*p1)-
     . (8*q(2)*q(7)*ek)-(4*q(2)*q(5)*ek)))+de*p1**2*(7*q(3)**2*p1**
     . 3+3*q(1)**2*p1**3+14*q(1)*q(6)*p1**2+20*q(1)*q(4)*p1**2-(8*q(2)*q
     . (3)*p1**2)-(6*q(3)*q(7)*p1)+14*q(4)*q(6)*p1+17*q(4)**2*p1+q(2)**2
     . *p1+6*q(2)*q(7))/6.0-(4.0/3.0*re*ek**3*p1**5*(3*(q(6)+q(4))**
     . 2*p1**2+(q(7)+q(5))**2))
c
c
      t0=-(1.0/3.0*he*p1*(12*q(1)*q(2)*p1**6-(6*q(1)*q(3)*ek*p1**5)-
     . (12*q(1)*q(7)*p1**5)-(12*q(2)*q(6)*p1**5)-(2*q(3)*q(6)*ek*p1**4)-
     . (8*q(3)*q(4)*ek*p1**4)+12*q(6)*q(7)*p1**4+12*q(1)*q(7)*ek*p1**3+
     . 16*q(2)*q(6)*ek*p1**3+6*q(1)*q(5)*ek*p1**3+16*q(2)*q(4)*ek*p1**3+
     . 3*q(3)*q(6)*ek**2*p1**2+3*q(3)*q(4)*ek**2*p1**2-(20*q(6)*q(7)*ek*
     . p1**2)-(8*q(4)*q(7)*ek*p1**2)-(6*q(5)*q(6)*ek*p1**2)-(q(1)*q(7)*
     . ek**2*p1)-(3*q(2)*q(6)*ek**2*p1)-(q(1)*q(5)*ek**2*p1)-(3*q(2)*q(4
     . )*ek**2*p1)-(q(4)*q(7)*ek**2)-(q(4)*q(5)*ek**2)))
      t0=t0+fe*p1*(14*q(1)*q(3)*p1**5-(2*q(3)*q(6)*p1**4)+12*q(3)*q(
     . 4)*p1**4-(5*q(1)*q(2)*p1**4)-(2*q(1)*q(3)*ek*p1**3)-(9*q(1)*q(7)*
     . p1**3)-(5*q(2)*q(6)*p1**3)-(10*q(2)*q(4)*p1**3)-(4*q(3)*q(6)*ek*
     . p1**2)-(6*q(3)*q(4)*ek*p1**2)+q(1)*q(2)*ek*p1**2+7*q(6)*q(7)*p1**
     . 2-(2*q(4)*q(7)*p1**2)+2*q(1)*q(7)*ek*p1+4*q(2)*q(6)*ek*p1+q(1)*q(
     . 5)*ek*p1+5*q(2)*q(4)*ek*p1+2*q(4)*q(7)*ek+q(4)*q(5)*ek)/3.0-(8.0/
     . 3.0*re*(q(6)+q(4))*(q(7)+q(5))*ek**3*p1**5)
      t0=t0-(1.0/3.0*ej*ek*p1**3*(8*q(1)*q(2)*p1**4-(16*q(1)*q(7)*p1
     . **3)-(16*q(2)*q(6)*p1**3)-(8*q(1)*q(5)*p1**3)-(8*q(2)*q(4)*p1**3)
     . +2*q(3)*q(6)*ek*p1**2+2*q(3)*q(4)*ek*p1**2+24*q(6)*q(7)*p1**2+8*q
     . (4)*q(7)*p1**2+8*q(5)*q(6)*p1**2+6*q(1)*q(7)*ek*p1+5*q(2)*q(6)*ek
     . *p1+6*q(1)*q(5)*ek*p1+5*q(2)*q(4)*ek*p1-(20*q(6)*q(7)*ek)-(14*q(4
     . )*q(7)*ek)-(13*q(5)*q(6)*ek)-(7*q(4)*q(5)*ek)))+2.0/3.0*el*ek
     . **2*p1**3*(4*q(1)*q(7)*p1**3+6*q(2)*q(6)*p1**3+4*q(1)*q(5)*p1**3+
     . 6*q(2)*q(4)*p1**3-(16*q(6)*q(7)*p1**2)-(12*q(4)*q(7)*p1**2)-(10*q
     . (5)*q(6)*p1**2)-(6*q(4)*q(5)*p1**2)+3*q(6)*q(7)*ek+3*q(4)*q(7)*ek
     . +3*q(5)*q(6)*ek+3*q(4)*q(5)*ek)
      bmn2(1,7)=t0-(1.0/3.0*de*p1*(2*q(1)*q(3)*p1**3+3*q(3)*q(6)*p1
     . **2+5*q(3)*q(4)*p1**2-(q(1)*q(2)*p1**2)-(q(1)*q(7)*p1)-(3*q(2)*q(
     . 6)*p1)-(4*q(2)*q(4)*p1)-(q(4)*q(7))))
c
c
      t0=he*(12*q(1)**2*p1**8-(24*q(1)*q(6)*p1**7)+12*q(6)**2*p1**6+
     . 36*q(2)**2*p1**6-(72*q(2)*q(7)*p1**5)+44*q(3)*q(7)*ek*p1**4-(12*q
     . (6)**2*ek*p1**4)-(24*q(4)*q(6)*ek*p1**4)+44*q(3)*q(5)*ek*p1**4-(
     . 12*q(4)**2*ek*p1**4)+36*q(7)**2*p1**4+q(6)**2*ek**2*p1**2+2*q(4)*
     . q(6)*ek**2*p1**2+q(4)**2*ek**2*p1**2-(44*q(7)**2*ek*p1**2)-(44*q(
     . 5)*q(7)*ek*p1**2)+7*q(7)**2*ek**2+14*q(5)*q(7)*ek**2+7*q(5)**2*ek
     . **2)/6.0
      t0=t0+fe*(32*q(3)**2*p1**6+27*q(1)**2*p1**6-(10*q(1)*q(6)*p1**
     . 5)+44*q(1)*q(4)*p1**5-(132*q(2)*q(3)*p1**5)+68*q(3)*q(7)*p1**4-(9
     . *q(6)**2*p1**4)-(28*q(4)*q(6)*p1**4)+8*q(4)**2*p1**4+81*q(2)**2*
     . p1**4+2*q(1)*q(6)*ek*p1**3+2*q(1)*q(4)*ek*p1**3-(30*q(2)*q(7)*p1
     . **3)-(16*q(3)*q(7)*ek*p1**2)+2*q(6)**2*ek*p1**2+6*q(4)*q(6)*ek*p1
     . **2-(16*q(3)*q(5)*ek*p1**2)+4*q(4)**2*ek*p1**2-(19*q(7)**2*p1**2)
     . +6*q(2)*q(7)*ek*p1+6*q(2)*q(5)*ek*p1+10*q(7)**2*ek+10*q(5)*q(7)*
     . ek)/6.0
      bmn2(2,2)=t0-(1.0/3.0*ej*ek*p1**2*(8*q(1)*q(6)*p1**5+8*q(1)*q(
     . 4)*p1**5-(8*q(6)**2*p1**4)-(8*q(4)*q(6)*p1**4)+24*q(2)*q(7)*p1**3
     . +24*q(2)*q(5)*p1**3+3*q(6)**2*ek*p1**2+6*q(4)*q(6)*ek*p1**2+3*q(4
     . )**2*ek*p1**2-(24*q(7)**2*p1**2)-(24*q(5)*q(7)*p1**2)+8*q(7)**2*
     . ek+16*q(5)*q(7)*ek+8*q(5)**2*ek))-(1.0/6.0*de*(5*q(3)**2*p1
     . **4+6*q(1)**2*p1**4-(2*q(1)*q(6)*p1**3)+10*q(1)*q(4)*p1**3-(30*q(
     . 2)*q(3)*p1**3)+20*q(3)*q(7)*p1**2-(q(6)**2*p1**2)-(4*q(4)*q(6)*p1
     . **2)+3*q(4)**2*p1**2+18*q(2)**2*p1**2-(6*q(2)*q(7)*p1)-(7*q(7)**2
     . )))+4.0/3.0*el*ek**2*p1**4*((q(6)+q(4))**2*p1**2+4*(q(7)+q(5)
     . )**2)+ae*(3*q(3)**2*p1**2+q(1)**2*p1**2-(3*q(2)*(2*q(3)*p1-q(
     . 2)))+2*q(1)*q(4)*p1+q(4)**2)/2.0
c
c
      t0=-(1.0/3.0*fe*p1*(11*q(1)**2*p1**6-(10*q(1)*q(6)*p1**5)+12*q
     . (1)*q(4)*p1**5-(32*q(2)*q(3)*p1**5)+32*q(3)*q(7)*p1**4-(q(6)**2*
     . p1**4)-(12*q(4)*q(6)*p1**4)+33*q(2)**2*p1**4+3*q(1)*q(6)*ek*p1**3
     . +3*q(1)*q(4)*ek*p1**3-(34*q(2)*q(7)*p1**3)-(13*q(3)*q(7)*ek*p1**2
     . )+q(6)**2*ek*p1**2+5*q(4)*q(6)*ek*p1**2-(13*q(3)*q(5)*ek*p1**2)+4
     . *q(4)**2*ek*p1**2+q(7)**2*p1**2+8*q(2)*q(7)*ek*p1+8*q(2)*q(5)*ek*
     . p1+5*q(7)**2*ek+5*q(5)*q(7)*ek))+he*ek*p1*(16*q(1)*q(6)*p1**5
     . +16*q(1)*q(4)*p1**5-(48*q(3)*q(7)*p1**4)+16*q(4)*q(6)*p1**4-(48*q
     . (3)*q(5)*p1**4)+16*q(4)**2*p1**4+44*q(2)*q(7)*p1**3+44*q(2)*q(5)*
     . p1**3-(q(6)**2*ek*p1**2)-(2*q(4)*q(6)*ek*p1**2)-(q(4)**2*ek*p1**2
     . )+4*q(7)**2*p1**2+4*q(5)*q(7)*p1**2-(7*q(7)**2*ek)-(14*q(5)*q(7)*
     . ek)-(7*q(5)**2*ek))/6.0
      bmn2(2,3)=t0-(1.0/6.0*de*p1*(12*q(3)**2*p1**4-(5*q(1)**2*p1**
     . 4)+8*q(1)*q(6)*p1**3-(2*q(1)*q(4)*p1**3)+10*q(2)*q(3)*p1**3-(34*q
     . (3)*q(7)*p1**2)+q(6)**2*p1**2+10*q(4)*q(6)*p1**2+4*q(4)**2*p1**2-
     . (15*q(2)**2*p1**2)+20*q(2)*q(7)*p1+7*q(7)**2))-(2.0/3.0*ej*(q
     . (7)+q(5))**2*ek**2*p1**3)-(1.0/2.0*ae*p1*(3*q(3)**2*p1**2+q(1
     . )**2*p1**2-(3*q(2)*(2*q(3)*p1-q(2)))+2*q(1)*q(4)*p1+q(4)**2))
c
c
      t0=-(1.0/3.0*fe*p1*(12*q(1)*q(3)*p1**5-(12*q(3)*q(6)*p1**4)-(
     . 22*q(1)*q(2)*p1**4)+3*q(1)*q(3)*ek*p1**3+10*q(1)*q(7)*p1**3+14*q(
     . 2)*q(6)*p1**3-(8*q(2)*q(4)*p1**3)+5*q(3)*q(6)*ek*p1**2+8*q(3)*q(4
     . )*ek*p1**2-(q(1)*q(2)*ek*p1**2)-(2*q(6)*q(7)*p1**2)+8*q(4)*q(7)*
     . p1**2-(5*q(1)*q(7)*ek*p1)-(3*q(2)*q(6)*ek*p1)-(3*q(1)*q(5)*ek*p1)
     . -(4*q(2)*q(4)*ek*p1)-(4*q(6)*q(7)*ek)-(9*q(4)*q(7)*ek)-(2*q(5)*q(
     . 6)*ek)-(5*q(4)*q(5)*ek)))
      t0=t0+he*ek*p1*(8*q(1)*q(3)*p1**5+8*q(3)*q(6)*p1**4+16*q(3)*q(
     . 4)*p1**4-(16*q(1)*q(7)*p1**3)-(12*q(2)*q(6)*p1**3)-(8*q(1)*q(5)*
     . p1**3)-(12*q(2)*q(4)*p1**3)-(q(3)*q(6)*ek*p1**2)-(q(3)*q(4)*ek*p1
     . **2)+4*q(6)*q(7)*p1**2-(12*q(4)*q(7)*p1**2)-(8*q(4)*q(5)*p1**2)+3
     . *q(1)*q(7)*ek*p1+q(2)*q(6)*ek*p1+3*q(1)*q(5)*ek*p1+q(2)*q(4)*ek*
     . p1+3*q(6)*q(7)*ek+6*q(4)*q(7)*ek+3*q(5)*q(6)*ek+6*q(4)*q(5)*ek)/
     . 3.0-(8.0/3.0*re*(q(6)+q(4))*(q(7)+q(5))*ek**3*p1**5)-(1.0/3.0
     . *ej*ek*p1**3*(8*q(1)*q(2)*p1**4-(8*q(1)*q(7)*p1**3)-(8*q(2)*q
     . (6)*p1**3)+8*q(6)*q(7)*p1**2+5*q(1)*q(7)*ek*p1+6*q(2)*q(6)*ek*p1+
     . 5*q(1)*q(5)*ek*p1+6*q(2)*q(4)*ek*p1-(15*q(6)*q(7)*ek)-(10*q(4)*q(
     . 7)*ek)-(9*q(5)*q(6)*ek)-(4*q(4)*q(5)*ek)))
      bmn2(2,4)=t0+2.0/3.0*el*ek**2*p1**3*(6*q(1)*q(7)*p1**3+4*q(2)*
     . q(6)*p1**3+6*q(1)*q(5)*p1**3+4*q(2)*q(4)*p1**3-(10*q(6)*q(7)*p1**
     . 2)-(4*q(4)*q(7)*p1**2)-(6*q(5)*q(6)*p1**2)+3*q(6)*q(7)*ek+3*q(4)*
     . q(7)*ek+3*q(5)*q(6)*ek+3*q(4)*q(5)*ek)+de*p1*(q(1)*q(3)*p1**
     . 3-(5*q(3)*q(6)*p1**2)-(4*q(3)*q(4)*p1**2)-(5*q(1)*q(2)*p1**2)+4*q
     . (1)*q(7)*p1+2*q(2)*q(6)*p1-(3*q(2)*q(4)*p1)+3*q(6)*q(7)+7*q(4)*q(
     . 7))/3.0-(ae*(q(1)*p1+q(4))*(q(3)*p1-q(2)))
c
c
      t0=-(1.0/3.0*ej*ek*p1*(4*q(1)**2*p1**6-(8*q(1)*q(6)*p1**5)+4*q
     . (6)**2*p1**4+12*q(2)**2*p1**4+5*q(1)*q(6)*ek*p1**3+5*q(1)*q(4)*ek
     . *p1**3-(24*q(2)*q(7)*p1**3)+4*q(3)*q(7)*ek*p1**2-(7*q(6)**2*ek*p1
     . **2)-(9*q(4)*q(6)*ek*p1**2)+4*q(3)*q(5)*ek*p1**2-(2*q(4)**2*ek*p1
     . **2)+12*q(7)**2*p1**2+16*q(2)*q(7)*ek*p1+16*q(2)*q(5)*ek*p1-(20*q
     . (7)**2*ek)-(20*q(5)*q(7)*ek)))+el*ek**2*p1*(12*q(1)*q(6)*p1**
     . 5+12*q(1)*q(4)*p1**5-(12*q(6)**2*p1**4)-(12*q(4)*q(6)*p1**4)+32*q
     . (2)*q(7)*p1**3+32*q(2)*q(5)*p1**3+3*q(6)**2*ek*p1**2+6*q(4)*q(6)*
     . ek*p1**2+3*q(4)**2*ek*p1**2-(32*q(7)**2*p1**2)-(32*q(5)*q(7)*p1**
     . 2)+9*q(7)**2*ek+18*q(5)*q(7)*ek+9*q(5)**2*ek)/3.0
      bmn2(2,5)=t0-(1.0/3.0*he*ek*(12*q(3)**2*p1**5+8*q(1)*q(6)*p1**
     . 4+8*q(1)*q(4)*p1**4-(22*q(2)*q(3)*p1**4)-(2*q(3)*q(7)*p1**3)-(4*q
     . (6)**2*p1**3)+4*q(4)**2*p1**3-(3*q(1)*q(6)*ek*p1**2)-(3*q(1)*q(4)
     . *ek*p1**2)+22*q(2)*q(7)*p1**2+7*q(3)*q(7)*ek*p1-(3*q(4)*q(6)*ek*
     . p1)+7*q(3)*q(5)*ek*p1-(3*q(4)**2*ek*p1)-(10*q(7)**2*p1)-(7*q(2)*q
     . (7)*ek)-(7*q(2)*q(5)*ek)))+fe*ek*(13*q(3)**2*p1**3+q(1)**2*p1
     . **3+4*q(1)*q(6)*p1**2+6*q(1)*q(4)*p1**2-(16*q(2)*q(3)*p1**2)-(10*
     . q(3)*q(7)*p1)+4*q(4)*q(6)*p1+5*q(4)**2*p1+3*q(2)**2*p1+10*q(2)*q(
     . 7))/6.0-(4.0/3.0*re*ek**3*p1**3*((q(6)+q(4))**2*p1**2+3*(q(7)
     . +q(5))**2))
c
c
      t0=-(1.0/3.0*he*p1*(12*q(1)*q(2)*p1**6-(8*q(1)*q(3)*ek*p1**5)-
     . (12*q(1)*q(7)*p1**5)-(12*q(2)*q(6)*p1**5)-(8*q(3)*q(4)*ek*p1**4)+
     . 12*q(6)*q(7)*p1**4+16*q(1)*q(7)*ek*p1**3+12*q(2)*q(6)*ek*p1**3+8*
     . q(1)*q(5)*ek*p1**3+12*q(2)*q(4)*ek*p1**3+q(3)*q(6)*ek**2*p1**2+q(
     . 3)*q(4)*ek**2*p1**2-(20*q(6)*q(7)*ek*p1**2)-(4*q(4)*q(7)*ek*p1**2
     . )-(8*q(5)*q(6)*ek*p1**2)-(3*q(1)*q(7)*ek**2*p1)-(q(2)*q(6)*ek**2*
     . p1)-(3*q(1)*q(5)*ek**2*p1)-(q(2)*q(4)*ek**2*p1)-(3*q(4)*q(7)*ek**
     . 2)-(3*q(4)*q(5)*ek**2)))
      t0=t0+fe*p1*(10*q(1)*q(3)*p1**5+2*q(3)*q(6)*p1**4+12*q(3)*q(4)
     . *p1**4-(5*q(1)*q(2)*p1**4)-(3*q(1)*q(3)*ek*p1**3)-(5*q(1)*q(7)*p1
     . **3)-(9*q(2)*q(6)*p1**3)-(14*q(2)*q(4)*p1**3)-(2*q(3)*q(6)*ek*p1
     . **2)-(5*q(3)*q(4)*ek*p1**2)+q(1)*q(2)*ek*p1**2+7*q(6)*q(7)*p1**2+
     . 2*q(4)*q(7)*p1**2+4*q(1)*q(7)*ek*p1+2*q(2)*q(6)*ek*p1+2*q(1)*q(5)
     . *ek*p1+3*q(2)*q(4)*ek*p1+4*q(4)*q(7)*ek+2*q(4)*q(5)*ek)/3.0-(8.0/
     . 3.0*re*(q(6)+q(4))*(q(7)+q(5))*ek**3*p1**5)-(1.0/3.0*ej*
     . ek*p1**3*(8*q(1)*q(2)*p1**4-(16*q(1)*q(7)*p1**3)-(16*q(2)*q(6)*p1
     . **3)-(8*q(1)*q(5)*p1**3)-(8*q(2)*q(4)*p1**3)+24*q(6)*q(7)*p1**2+8
     . *q(4)*q(7)*p1**2+8*q(5)*q(6)*p1**2+5*q(1)*q(7)*ek*p1+6*q(2)*q(6)*
     . ek*p1+5*q(1)*q(5)*ek*p1+6*q(2)*q(4)*ek*p1-(20*q(6)*q(7)*ek)-(15*q
     . (4)*q(7)*ek)-(14*q(5)*q(6)*ek)-(9*q(4)*q(5)*ek)))
      bmn2(2,6)=t0+2.0/3.0*el*ek**2*p1**3*(6*q(1)*q(7)*p1**3+4*q(2)*
     . q(6)*p1**3+6*q(1)*q(5)*p1**3+4*q(2)*q(4)*p1**3-(16*q(6)*q(7)*p1**
     . 2)-(10*q(4)*q(7)*p1**2)-(12*q(5)*q(6)*p1**2)-(6*q(4)*q(5)*p1**2)+
     . 3*q(6)*q(7)*ek+3*q(4)*q(7)*ek+3*q(5)*q(6)*ek+3*q(4)*q(5)*ek)-(1.0
     . /3.0*de*p1*(4*q(1)*q(3)*p1**3+q(3)*q(6)*p1**2+5*q(3)*q(4)*p1
     . **2-(q(1)*q(2)*p1**2)-(3*q(1)*q(7)*p1)-(q(2)*q(6)*p1)-(2*q(2)*q(4
     . )*p1)-(3*q(4)*q(7))))
c
c
      t0=-(1.0/3.0*he*(6*q(1)**2*p1**7-(12*q(1)*q(6)*p1**6)+12*q(3)
     . **2*ek*p1**5+6*q(6)**2*p1**5+18*q(2)**2*p1**5+16*q(1)*q(6)*ek*p1
     . **4+16*q(1)*q(4)*ek*p1**4-(22*q(2)*q(3)*ek*p1**4)-(36*q(2)*q(7)*
     . p1**4)-(4*q(3)*q(7)*ek*p1**3)-(10*q(6)**2*ek*p1**3)-(4*q(4)*q(6)*
     . ek*p1**3)-(2*q(3)*q(5)*ek*p1**3)+6*q(4)**2*ek*p1**3+18*q(7)**2*p1
     . **3-(3*q(1)*q(6)*ek**2*p1**2)-(3*q(1)*q(4)*ek**2*p1**2)+44*q(2)*q
     . (7)*ek*p1**2+22*q(2)*q(5)*ek*p1**2+7*q(3)*q(7)*ek**2*p1-(3*q(4)*q
     . (6)*ek**2*p1)+7*q(3)*q(5)*ek**2*p1-(3*q(4)**2*ek**2*p1)-(30*q(7)
     . **2*ek*p1)-(20*q(5)*q(7)*ek*p1)-(7*q(2)*q(7)*ek**2)-(7*q(2)*q(5)*
     . ek**2)))
      t0=t0-(1.0/3.0*ej*ek*p1*(4*q(1)**2*p1**6-(16*q(1)*q(6)*p1**5)-
     . (8*q(1)*q(4)*p1**5)+12*q(6)**2*p1**4+8*q(4)*q(6)*p1**4+12*q(2)**2
     . *p1**4+5*q(1)*q(6)*ek*p1**3+5*q(1)*q(4)*ek*p1**3-(48*q(2)*q(7)*p1
     . **3)-(24*q(2)*q(5)*p1**3)+4*q(3)*q(7)*ek*p1**2-(10*q(6)**2*ek*p1
     . **2)-(15*q(4)*q(6)*ek*p1**2)+4*q(3)*q(5)*ek*p1**2-(5*q(4)**2*ek*
     . p1**2)+36*q(7)**2*p1**2+24*q(5)*q(7)*p1**2+16*q(2)*q(7)*ek*p1+16*
     . q(2)*q(5)*ek*p1-(30*q(7)**2*ek)-(40*q(5)*q(7)*ek)-(10*q(5)**2*ek)
     . ))
      t0=t0+el*ek**2*p1*(12*q(1)*q(6)*p1**5+12*q(1)*q(4)*p1**5-(16*q
     . (6)**2*p1**4)-(20*q(4)*q(6)*p1**4)-(4*q(4)**2*p1**4)+32*q(2)*q(7)
     . *p1**3+32*q(2)*q(5)*p1**3+3*q(6)**2*ek*p1**2+6*q(4)*q(6)*ek*p1**2
     . +3*q(4)**2*ek*p1**2-(48*q(7)**2*p1**2)-(64*q(5)*q(7)*p1**2)-(16*q
     . (5)**2*p1**2)+9*q(7)**2*ek+18*q(5)*q(7)*ek+9*q(5)**2*ek)/3.0
      bmn2(2,7)=t0-(1.0/6.0*fe*(32*q(3)**2*p1**5+5*q(1)**2*p1**5+10*
     . q(1)*q(6)*p1**4+20*q(1)*q(4)*p1**4-(68*q(2)*q(3)*p1**4)-(13*q(3)
     . **2*ek*p1**3)-(q(1)**2*ek*p1**3)+4*q(3)*q(7)*p1**3-(7*q(6)**2*p1
     . **3)-(4*q(4)*q(6)*p1**3)+8*q(4)**2*p1**3+15*q(2)**2*p1**3-(8*q(1)
     . *q(6)*ek*p1**2)-(10*q(1)*q(4)*ek*p1**2)+16*q(2)*q(3)*ek*p1**2+38*
     . q(2)*q(7)*p1**2+20*q(3)*q(7)*ek*p1-(8*q(4)*q(6)*ek*p1)+10*q(3)*q(
     . 5)*ek*p1-(9*q(4)**2*ek*p1)-(3*q(2)**2*ek*p1)-(21*q(7)**2*p1)-(20*
     . q(2)*q(7)*ek)-(10*q(2)*q(5)*ek)))+de*(17*q(3)**2*p1**3+q(1)
     . **2*p1**3+6*q(1)*q(6)*p1**2+8*q(1)*q(4)*p1**2-(20*q(2)*q(3)*p1**2
     . )-(14*q(3)*q(7)*p1)+6*q(4)*q(6)*p1+7*q(4)**2*p1+3*q(2)**2*p1+14*q
     . (2)*q(7))/6.0-(4.0/3.0*re*ek**3*p1**3*((q(6)+q(4))**2*p1**2+3
     . *(q(7)+q(5))**2))
c
c
      t0=fe*p1**2*(4*q(1)**2*p1**6-(8*q(1)*q(6)*p1**5)+4*q(6)**2*p1
     . **4+16*q(2)**2*p1**4+5*q(1)*q(6)*ek*p1**3+5*q(1)*q(4)*ek*p1**3-(
     . 32*q(2)*q(7)*p1**3)-(18*q(3)*q(7)*ek*p1**2)+q(6)**2*ek*p1**2+7*q(
     . 4)*q(6)*ek*p1**2-(18*q(3)*q(5)*ek*p1**2)+6*q(4)**2*ek*p1**2+16*q(
     . 7)**2*p1**2+13*q(2)*q(7)*ek*p1+13*q(2)*q(5)*ek*p1+5*q(7)**2*ek+5*
     . q(5)*q(7)*ek)/3.0-(1.0/6.0*he*ek*p1**2*(16*q(1)*q(6)*p1**5+16
     . *q(1)*q(4)*p1**5-(16*q(6)**2*p1**4)-(16*q(4)*q(6)*p1**4)+48*q(2)*
     . q(7)*p1**3+48*q(2)*q(5)*p1**3-(q(6)**2*ek*p1**2)-(2*q(4)*q(6)*ek*
     . p1**2)-(q(4)**2*ek*p1**2)-(48*q(7)**2*p1**2)-(48*q(5)*q(7)*p1**2)
     . -(7*q(7)**2*ek)-(14*q(5)*q(7)*ek)-(7*q(5)**2*ek)))
      bmn2(3,3)=t0+de*p1**2*(36*q(3)**2*p1**4-(3*q(1)**2*p1**4)+14*
     . q(1)*q(6)*p1**3+8*q(1)*q(4)*p1**3-(24*q(2)*q(3)*p1**3)-(48*q(3)*q
     . (7)*p1**2)+q(6)**2*p1**2+16*q(4)*q(6)*p1**2+12*q(4)**2*p1**2-(5*q
     . (2)**2*p1**2)+34*q(2)*q(7)*p1+7*q(7)**2)/6.0+4.0/3.0*ej*ek**2
     . *p1**4*((q(6)+q(4))**2*p1**2+4*(q(7)+q(5))**2)+ae*p1**2*(3*q(
     . 3)**2*p1**2+q(1)**2*p1**2-(3*q(2)*(2*q(3)*p1-q(2)))+2*q(1)*q(4)*
     . p1+q(4)**2)/2.0
c
c
      t0=-(1.0/3.0*he*ek*p1**2*(8*q(1)*q(3)*p1**5-(8*q(3)*q(6)*p1**4
     . )-(8*q(1)*q(2)*p1**4)-(8*q(1)*q(7)*p1**3)-(8*q(2)*q(6)*p1**3)-(8*
     . q(1)*q(5)*p1**3)-(16*q(2)*q(4)*p1**3)-(q(3)*q(6)*ek*p1**2)-(q(3)*
     . q(4)*ek*p1**2)+24*q(6)*q(7)*p1**2+16*q(4)*q(7)*p1**2+8*q(5)*q(6)*
     . p1**2+3*q(1)*q(7)*ek*p1+q(2)*q(6)*ek*p1+3*q(1)*q(5)*ek*p1+q(2)*q(
     . 4)*ek*p1+3*q(6)*q(7)*ek+6*q(4)*q(7)*ek+3*q(5)*q(6)*ek+6*q(4)*q(5)
     . *ek))-(1.0/3.0*fe*p1**2*(12*q(1)*q(2)*p1**4-(5*q(1)*q(3)*ek*
     . p1**3)-(12*q(1)*q(7)*p1**3)-(12*q(2)*q(6)*p1**3)-(7*q(3)*q(6)*ek*
     . p1**2)-(12*q(3)*q(4)*ek*p1**2)+3*q(1)*q(2)*ek*p1**2+12*q(6)*q(7)*
     . p1**2+6*q(1)*q(7)*ek*p1+5*q(2)*q(6)*ek*p1+4*q(1)*q(5)*ek*p1+8*q(2
     . )*q(4)*ek*p1+4*q(6)*q(7)*ek+10*q(4)*q(7)*ek+2*q(5)*q(6)*ek+6*q(4)
     . *q(5)*ek))
      bmn2(3,4)=t0-(8.0/3.0*el*(q(6)+q(4))*(q(7)+q(5))*ek**3*p1**4)+
     . de*p1**2*(4*q(1)*q(3)*p1**3+8*q(3)*q(6)*p1**2+12*q(3)*q(4)*
     . p1**2+q(1)*q(2)*p1**2-(5*q(1)*q(7)*p1)-(5*q(2)*q(6)*p1)-(4*q(2)*q
     . (4)*p1)-(3*q(6)*q(7))-(8*q(4)*q(7)))/3.0+2.0/3.0*ej*ek**2*p1
     . **4*(4*q(3)*q(6)*p1**2+4*q(3)*q(4)*p1**2-(q(1)*q(7)*p1)-(q(1)*q(5
     . )*p1)-(15*q(6)*q(7))-(16*q(4)*q(7))-(11*q(5)*q(6))-(12*q(4)*q(5))
     . )+ae*p1*(q(1)*p1+q(4))*(q(3)*p1-q(2))
c
c
      t0=he*ek*p1*(3*q(1)**2*p1**5+2*q(1)*q(6)*p1**4+8*q(1)*q(4)*p1
     . **4-(24*q(2)*q(3)*p1**4)+24*q(3)*q(7)*p1**3-(5*q(6)**2*p1**3)-(8*
     . q(4)*q(6)*p1**3)+11*q(2)**2*p1**3-(3*q(1)*q(6)*ek*p1**2)-(3*q(1)*
     . q(4)*ek*p1**2)+2*q(2)*q(7)*p1**2+7*q(3)*q(7)*ek*p1-(3*q(4)*q(6)*
     . ek*p1)+7*q(3)*q(5)*ek*p1-(3*q(4)**2*ek*p1)-(13*q(7)**2*p1)-(7*q(2
     . )*q(7)*ek)-(7*q(2)*q(5)*ek))/3.0-(2.0/3.0*ej*ek**2*p1**2*(q(1
     . )*q(6)*p1**3+q(1)*q(4)*p1**3-(16*q(3)*q(7)*p1**2)+5*q(6)**2*p1**2
     . +11*q(4)*q(6)*p1**2-(16*q(3)*q(5)*p1**2)+6*q(4)**2*p1**2+2*q(2)*q
     . (7)*p1+2*q(2)*q(5)*p1+14*q(7)**2+14*q(5)*q(7)))
      bmn2(3,5)=t0-(1.0/3.0*fe*ek*p1*(9*q(3)**2*p1**3+q(1)**2*p1**3+
     . 2*q(1)*q(6)*p1**2+4*q(1)*q(4)*p1**2-(13*q(2)*q(3)*p1**2)-(5*q(3)*
     . q(7)*p1)+2*q(4)*q(6)*p1+3*q(4)**2*p1+4*q(2)**2*p1+5*q(2)*q(7)))-(
     . 4.0/3.0*el*ek**3*p1**2*((q(6)+q(4))**2*p1**2+3*(q(7)+q(5))**2
     . ))
c
c
      t0=-(1.0/3.0*fe*p1**2*(8*q(1)*q(3)*p1**5-(8*q(3)*q(6)*p1**4)-(
     . 10*q(1)*q(2)*p1**4)-(5*q(1)*q(3)*ek*p1**3)+2*q(1)*q(7)*p1**3-(2*q
     . (2)*q(6)*p1**3)-(12*q(2)*q(4)*p1**3)-(2*q(3)*q(6)*ek*p1**2)-(7*q(
     . 3)*q(4)*ek*p1**2)+3*q(1)*q(2)*ek*p1**2+10*q(6)*q(7)*p1**2+12*q(4)
     . *q(7)*p1**2+4*q(1)*q(7)*ek*p1+2*q(2)*q(6)*ek*p1+2*q(1)*q(5)*ek*p1
     . +5*q(2)*q(4)*ek*p1+4*q(4)*q(7)*ek+2*q(4)*q(5)*ek))-(1.0/3.0*he
     . *ek*p1**2*(8*q(1)*q(3)*p1**5-(16*q(3)*q(6)*p1**4)-(8*q(3)*q(4)*
     . p1**4)-(8*q(1)*q(2)*p1**4)-(2*q(1)*q(7)*p1**3)-(2*q(1)*q(5)*p1**3
     . )-(8*q(2)*q(4)*p1**3)-(q(3)*q(6)*ek*p1**2)-(q(3)*q(4)*ek*p1**2)+
     . 26*q(6)*q(7)*p1**2+24*q(4)*q(7)*p1**2+10*q(5)*q(6)*p1**2+8*q(4)*q
     . (5)*p1**2+3*q(1)*q(7)*ek*p1+q(2)*q(6)*ek*p1+3*q(1)*q(5)*ek*p1+q(2
     . )*q(4)*ek*p1+3*q(4)*q(7)*ek+3*q(4)*q(5)*ek))
      bmn2(3,6)=t0-(8.0/3.0*el*(q(6)+q(4))*(q(7)+q(5))*ek**3*p1**4)+
     . de*p1**2*(7*q(1)*q(3)*p1**3+q(3)*q(6)*p1**2+8*q(3)*q(4)*p1**
     . 2-(4*q(1)*q(2)*p1**2)-(3*q(1)*q(7)*p1)-(q(2)*q(6)*p1)-(5*q(2)*q(4
     . )*p1)-(3*q(4)*q(7)))/3.0+2.0/3.0*ej*ek**2*p1**4*(4*q(3)*q(6)*
     . p1**2+4*q(3)*q(4)*p1**2-(q(1)*q(7)*p1)-(q(1)*q(5)*p1)-(14*q(6)*q(
     . 7))-(15*q(4)*q(7))-(10*q(5)*q(6))-(11*q(4)*q(5)))
c
c
      t0=fe*p1*(7*q(1)**2*p1**5-(2*q(1)*q(6)*p1**4)+12*q(1)*q(4)*p1
     . **4-(32*q(2)*q(3)*p1**4)-(9*q(3)**2*ek*p1**3)-(q(1)**2*ek*p1**3)+
     . 32*q(3)*q(7)*p1**3-(5*q(6)**2*p1**3)-(12*q(4)*q(6)*p1**3)+17*q(2)
     . **2*p1**3-(4*q(1)*q(6)*ek*p1**2)-(6*q(1)*q(4)*ek*p1**2)+13*q(2)*q
     . (3)*ek*p1**2-(2*q(2)*q(7)*p1**2)+10*q(3)*q(7)*ek*p1-(4*q(4)*q(6)*
     . ek*p1)+5*q(3)*q(5)*ek*p1-(5*q(4)**2*ek*p1)-(4*q(2)**2*ek*p1)-(15*
     . q(7)**2*p1)-(10*q(2)*q(7)*ek)-(5*q(2)*q(5)*ek))/3.0
      t0=t0+he*ek*p1*(3*q(1)**2*p1**5+2*q(1)*q(6)*p1**4+8*q(1)*q(4)*
     . p1**4-(24*q(2)*q(3)*p1**4)+48*q(3)*q(7)*p1**3-(13*q(6)**2*p1**3)-
     . (24*q(4)*q(6)*p1**3)+24*q(3)*q(5)*p1**3-(8*q(4)**2*p1**3)+11*q(2)
     . **2*p1**3-(3*q(1)*q(6)*ek*p1**2)-(3*q(1)*q(4)*ek*p1**2)+4*q(2)*q(
     . 7)*p1**2+2*q(2)*q(5)*p1**2+7*q(3)*q(7)*ek*p1-(3*q(4)*q(6)*ek*p1)+
     . 7*q(3)*q(5)*ek*p1-(3*q(4)**2*ek*p1)-(39*q(7)**2*p1)-(26*q(5)*q(7)
     . *p1)-(7*q(2)*q(7)*ek)-(7*q(2)*q(5)*ek))/3.0-(2.0/3.0*ej*ek**2
     . *p1**2*(q(1)*q(6)*p1**3+q(1)*q(4)*p1**3-(16*q(3)*q(7)*p1**2)+7*q(
     . 6)**2*p1**2+15*q(4)*q(6)*p1**2-(16*q(3)*q(5)*p1**2)+8*q(4)**2*p1
     . **2+2*q(2)*q(7)*p1+2*q(2)*q(5)*p1+21*q(7)**2+28*q(5)*q(7)+7*q(5)
     . **2))
      bmn2(3,7)=t0-(1.0/3.0*de*p1*(12*q(3)**2*p1**3+q(1)**2*p1**3+3
     . *q(1)*q(6)*p1**2+5*q(1)*q(4)*p1**2-(17*q(2)*q(3)*p1**2)-(7*q(3)*q
     . (7)*p1)+3*q(4)*q(6)*p1+4*q(4)**2*p1+5*q(2)**2*p1+7*q(2)*q(7)))-(
     . 4.0/3.0*el*ek**3*p1**2*((q(6)+q(4))**2*p1**2+3*(q(7)+q(5))**2
     . ))
c
c
      t0=-(1.0/6.0*he*ek*(44*q(1)**2*p1**6+56*q(1)*q(6)*p1**5+144*q(
     . 1)*q(4)*p1**5-(32*q(2)*q(3)*p1**5)-(q(3)**2*ek*p1**4)-(7*q(1)**2*
     . ek*p1**4)+32*q(3)*q(7)*p1**4-(100*q(6)**2*p1**4)-(144*q(4)*q(6)*
     . p1**4)+12*q(2)**2*p1**4-(28*q(1)*q(6)*ek*p1**3)-(42*q(1)*q(4)*ek*
     . p1**3)+2*q(2)*q(3)*ek*p1**3+24*q(2)*q(7)*p1**3+16*q(2)*q(5)*p1**3
     . +12*q(3)*q(7)*ek*p1**2-(7*q(6)**2*ek*p1**2)-(42*q(4)*q(6)*ek*p1**
     . 2)+12*q(3)*q(5)*ek*p1**2-(42*q(4)**2*ek*p1**2)-(q(2)**2*ek*p1**2)
     . -(36*q(7)**2*p1**2)-(16*q(5)*q(7)*p1**2)-(12*q(2)*q(7)*ek*p1)-(12
     . *q(2)*q(5)*ek*p1)-(q(7)**2*ek)-(2*q(5)*q(7)*ek)-(q(5)**2*ek)))
      t0=t0+fe*(16*q(1)**2*p1**6-(32*q(1)*q(6)*p1**5)+6*q(3)**2*ek*
     . p1**4+8*q(1)**2*ek*p1**4+16*q(6)**2*p1**4+4*q(2)**2*p1**4+23*q(1)
     . *q(6)*ek*p1**3+39*q(1)*q(4)*ek*p1**3-(8*q(2)*q(3)*ek*p1**3)-(8*q(
     . 2)*q(7)*p1**3)-(10*q(3)*q(7)*ek*p1**2)+5*q(6)**2*ek*p1**2+33*q(4)
     . *q(6)*ek*p1**2-(6*q(3)*q(5)*ek*p1**2)+36*q(4)**2*ek*p1**2+2*q(2)
     . **2*ek*p1**2+4*q(7)**2*p1**2+9*q(2)*q(7)*ek*p1+5*q(2)*q(5)*ek*p1+
     . q(7)**2*ek+q(5)*q(7)*ek)/3.0
      t0=t0+2.0/3.0*el*ek**2*p1**2*(8*q(1)**2*p1**6-(16*q(1)*q(6)*p1
     . **5)+8*q(6)**2*p1**4+2*q(2)**2*p1**4+9*q(1)*q(6)*ek*p1**3+9*q(1)*
     . q(4)*ek*p1**3-(4*q(2)*q(7)*p1**3)-(4*q(3)*q(7)*ek*p1**2)+15*q(6)
     . **2*ek*p1**2+39*q(4)*q(6)*ek*p1**2-(4*q(3)*q(5)*ek*p1**2)+24*q(4)
     . **2*ek*p1**2+2*q(7)**2*p1**2+3*q(2)*q(7)*ek*p1+3*q(2)*q(5)*ek*p1+
     . 5*q(7)**2*ek+9*q(5)*q(7)*ek+4*q(5)**2*ek)-(1.0/6.0*re*ek**3*
     . p1**2*(48*q(1)*q(6)*p1**5+48*q(1)*q(4)*p1**5-(48*q(6)**2*p1**4)-(
     . 48*q(4)*q(6)*p1**4)+16*q(2)*q(7)*p1**3+16*q(2)*q(5)*p1**3-(9*q(6)
     . **2*ek*p1**2)-(18*q(4)*q(6)*ek*p1**2)-(9*q(4)**2*ek*p1**2)-(16*q(
     . 7)**2*p1**2)-(16*q(5)*q(7)*p1**2)-(3*q(7)**2*ek)-(6*q(5)*q(7)*ek)
     . -(3*q(5)**2*ek)))
      bmn2(4,4)=t0+de*(12*q(3)**2*p1**4-(5*q(1)**2*p1**4)+34*q(1)*q
     . (6)*p1**3+24*q(1)*q(4)*p1**3-(8*q(2)*q(3)*p1**3)-(16*q(3)*q(7)*p1
     . **2)+7*q(6)**2*p1**2+48*q(4)*q(6)*p1**2+36*q(4)**2*p1**2-(3*q(2)
     . **2*p1**2)+14*q(2)*q(7)*p1+q(7)**2)/6.0+ej*ek**2*p1**2*(4*q(3
     . )**2*p1**4-(8*q(1)**2*p1**4)+28*q(1)*q(6)*p1**3+12*q(1)*q(4)*p1**
     . 3-(32*q(3)*q(7)*p1**2)+76*q(6)**2*p1**2+180*q(4)*q(6)*p1**2-(24*q
     . (3)*q(5)*p1**2)+96*q(4)**2*p1**2-(3*q(2)**2*p1**2)+10*q(2)*q(7)*
     . p1+4*q(2)*q(5)*p1+25*q(7)**2+28*q(5)*q(7)+4*q(5)**2)/3.0+2*te
     . *ek**4*p1**4*(3*(q(6)+q(4))**2*p1**2+(q(7)+q(5))**2)+ae*(q(3)
     . **2*p1**2+3*q(1)**2*p1**2-(q(2)*(2*q(3)*p1-q(2)))+6*q(1)*q(4)*p1+
     . 3*q(4)**2)/2.0
c
c
      t0=he*ek*(8*q(1)*q(3)*p1**5-(8*q(3)*q(6)*p1**4)-(8*q(1)*q(2)*
     . p1**4)-(3*q(1)*q(3)*ek*p1**3)-(8*q(2)*q(4)*p1**3)-(3*q(3)*q(6)*ek
     . *p1**2)-(6*q(3)*q(4)*ek*p1**2)+3*q(1)*q(2)*ek*p1**2+8*q(6)*q(7)*
     . p1**2+8*q(4)*q(7)*p1**2+q(1)*q(7)*ek*p1+3*q(2)*q(6)*ek*p1+q(1)*q(
     . 5)*ek*p1+6*q(2)*q(4)*ek*p1+q(4)*q(7)*ek+q(4)*q(5)*ek)/3.0+2.0/3.0
     . *el*ek**2*p1**2*(6*q(1)*q(2)*p1**4-(6*q(1)*q(7)*p1**3)-(6*q(2
     . )*q(6)*p1**3)-(4*q(3)*q(6)*ek*p1**2)-(4*q(3)*q(4)*ek*p1**2)+6*q(6
     . )*q(7)*p1**2+3*q(1)*q(7)*ek*p1+3*q(2)*q(6)*ek*p1+3*q(1)*q(5)*ek*
     . p1+3*q(2)*q(4)*ek*p1+6*q(6)*q(7)*ek+9*q(4)*q(7)*ek+5*q(5)*q(6)*ek
     . +8*q(4)*q(5)*ek)+4*te*(q(6)+q(4))*(q(7)+q(5))*ek**4*p1**4
      bmn2(4,5)=t0-(1.0/3.0*re*ek**3*p1**2*(8*q(1)*q(7)*p1**3+8*q(2)
     . *q(6)*p1**3+8*q(1)*q(5)*p1**3+8*q(2)*q(4)*p1**3-(16*q(6)*q(7)*p1
     . **2)-(8*q(4)*q(7)*p1**2)-(8*q(5)*q(6)*p1**2)-(3*q(6)*q(7)*ek)-(3*
     . q(4)*q(7)*ek)-(3*q(5)*q(6)*ek)-(3*q(4)*q(5)*ek)))-(1.0/3.0*fe
     . *ek*(4*q(1)*q(3)*p1**3+2*q(3)*q(6)*p1**2+6*q(3)*q(4)*p1**2-(3*q(1
     . )*q(2)*p1**2)-(q(1)*q(7)*p1)-(2*q(2)*q(6)*p1)-(5*q(2)*q(4)*p1)-(q
     . (4)*q(7))))-(1.0/3.0*ej*ek**2*p1**2*(2*q(1)*q(3)*p1**3+22*q(3
     . )*q(6)*p1**2+24*q(3)*q(4)*p1**2+5*q(1)*q(2)*p1**2-(7*q(1)*q(7)*p1
     . )-(9*q(2)*q(6)*p1)-(4*q(2)*q(4)*p1)-(21*q(6)*q(7))-(28*q(4)*q(7))
     . -(8*q(5)*q(6))-(8*q(4)*q(5))))
c
c
      t0=el*ek**2*p1**2*(16*q(1)**2*p1**6-(64*q(1)*q(6)*p1**5)-(32*q
     . (1)*q(4)*p1**5)+48*q(6)**2*p1**4+32*q(4)*q(6)*p1**4+4*q(2)**2*p1
     . **4+18*q(1)*q(6)*ek*p1**3+18*q(1)*q(4)*ek*p1**3-(20*q(2)*q(7)*p1
     . **3)-(12*q(2)*q(5)*p1**3)-(8*q(3)*q(7)*ek*p1**2)+21*q(6)**2*ek*p1
     . **2+60*q(4)*q(6)*ek*p1**2-(8*q(3)*q(5)*ek*p1**2)+39*q(4)**2*ek*p1
     . **2+16*q(7)**2*p1**2+12*q(5)*q(7)*p1**2+6*q(2)*q(7)*ek*p1+6*q(2)*
     . q(5)*ek*p1+7*q(7)**2*ek+12*q(5)*q(7)*ek+5*q(5)**2*ek)/3.0
      t0=t0+ej*ek*p1**2*(12*q(1)**2*p1**6-(24*q(1)*q(6)*p1**5)+4*q(3
     . )**2*ek*p1**4-(8*q(1)**2*ek*p1**4)+12*q(6)**2*p1**4+4*q(2)**2*p1
     . **4+44*q(1)*q(6)*ek*p1**3+28*q(1)*q(4)*ek*p1**3-(8*q(2)*q(7)*p1**
     . 3)-(30*q(3)*q(7)*ek*p1**2)+54*q(6)**2*ek*p1**2+152*q(4)*q(6)*ek*
     . p1**2-(22*q(3)*q(5)*ek*p1**2)+90*q(4)**2*ek*p1**2-(3*q(2)**2*ek*
     . p1**2)+4*q(7)**2*p1**2+15*q(2)*q(7)*ek*p1+9*q(2)*q(5)*ek*p1+18*q(
     . 7)**2*ek+21*q(5)*q(7)*ek+4*q(5)**2*ek)/3.0
      t0=t0-(1.0/6.0*re*ek**3*p1**2*(48*q(1)*q(6)*p1**5+48*q(1)*q(4)
     . *p1**5-(72*q(6)**2*p1**4)-(96*q(4)*q(6)*p1**4)-(24*q(4)**2*p1**4)
     . +16*q(2)*q(7)*p1**3+16*q(2)*q(5)*p1**3-(9*q(6)**2*ek*p1**2)-(18*q
     . (4)*q(6)*ek*p1**2)-(9*q(4)**2*ek*p1**2)-(24*q(7)**2*p1**2)-(32*q(
     . 5)*q(7)*p1**2)-(8*q(5)**2*p1**2)-(3*q(7)**2*ek)-(6*q(5)*q(7)*ek)-
     . (3*q(5)**2*ek)))
      t0=t0+he*ek*p1*(8*q(3)**2*p1**5-(44*q(1)**2*p1**5)+32*q(1)*q(6
     . )*p1**4-(56*q(1)*q(4)*p1**4)+16*q(2)*q(3)*p1**4+q(3)**2*ek*p1**3+
     . 7*q(1)**2*ek*p1**3-(48*q(3)*q(7)*p1**3)+84*q(6)**2*p1**3+200*q(4)
     . *q(6)*p1**3-(16*q(3)*q(5)*p1**3)+72*q(4)**2*p1**3-(12*q(2)**2*p1
     . **3)+14*q(1)*q(6)*ek*p1**2+28*q(1)*q(4)*ek*p1**2-(2*q(2)*q(3)*ek*
     . p1**2)+8*q(2)*q(7)*p1**2-(6*q(3)*q(7)*ek*p1)+14*q(4)*q(6)*ek*p1-(
     . 6*q(3)*q(5)*ek*p1)+21*q(4)**2*ek*p1+q(2)**2*ek*p1+28*q(7)**2*p1+
     . 16*q(5)*q(7)*p1+6*q(2)*q(7)*ek+6*q(2)*q(5)*ek)/6.0
      bmn2(4,6)=t0-(1.0/6.0*fe*p1*(34*q(1)**2*p1**5-(4*q(1)*q(6)*p1
     . **4)+64*q(1)*q(4)*p1**4-(24*q(2)*q(3)*p1**4)-(7*q(3)**2*ek*p1**3)
     . -(13*q(1)**2*ek*p1**3)+24*q(3)*q(7)*p1**3-(30*q(6)**2*p1**3)-(64*
     . q(4)*q(6)*p1**3)+14*q(2)**2*p1**3-(20*q(1)*q(6)*ek*p1**2)-(46*q(1
     . )*q(4)*ek*p1**2)+10*q(2)*q(3)*ek*p1**2-(4*q(2)*q(7)*p1**2)+8*q(3)
     . *q(7)*ek*p1-(20*q(4)*q(6)*ek*p1)+4*q(3)*q(5)*ek*p1-(33*q(4)**2*ek
     . *p1)-(3*q(2)**2*ek*p1)-(10*q(7)**2*p1)-(8*q(2)*q(7)*ek)-(4*q(2)*q
     . (5)*ek)))+de*p1*(4*q(3)**2*p1**3+5*q(1)**2*p1**3+7*q(1)*q(6)
     . *p1**2+17*q(1)*q(4)*p1**2-(5*q(2)*q(3)*p1**2)-(3*q(3)*q(7)*p1)+7*
     . q(4)*q(6)*p1+12*q(4)**2*p1+q(2)**2*p1+3*q(2)*q(7))/3.0+2*te*
     . ek**4*p1**4*(3*(q(6)+q(4))**2*p1**2+(q(7)+q(5))**2)
c
c
       t0=fe*(12*q(1)*q(3)*p1**5-(12*q(3)*q(6)*p1**4)-(10*q(1)*q(2)*
     . p1**4)-(6*q(1)*q(3)*ek*p1**3)-(2*q(1)*q(7)*p1**3)+2*q(2)*q(6)*p1
     . **3-(8*q(2)*q(4)*p1**3)-(4*q(3)*q(6)*ek*p1**2)-(10*q(3)*q(4)*ek*
     . p1**2)+5*q(1)*q(2)*ek*p1**2+10*q(6)*q(7)*p1**2+8*q(4)*q(7)*p1**2+
     . 2*q(1)*q(7)*ek*p1+4*q(2)*q(6)*ek*p1+q(1)*q(5)*ek*p1+9*q(2)*q(4)*
     . ek*p1+2*q(4)*q(7)*ek+q(4)*q(5)*ek)/3.0
       t0=t0+he*ek*(8*q(1)*q(3)*p1**5-(24*q(3)*q(6)*p1**4)-(16*q(3)*q
     . (4)*p1**4)-(16*q(1)*q(2)*p1**4)-(3*q(1)*q(3)*ek*p1**3)+8*q(1)*q(7
     . )*p1**3+4*q(2)*q(6)*p1**3-(12*q(2)*q(4)*p1**3)-(3*q(3)*q(6)*ek*p1
     . **2)-(6*q(3)*q(4)*ek*p1**2)+3*q(1)*q(2)*ek*p1**2+28*q(6)*q(7)*p1
     . **2+36*q(4)*q(7)*p1**2+8*q(5)*q(6)*p1**2+8*q(4)*q(5)*p1**2+q(1)*q
     . (7)*ek*p1+3*q(2)*q(6)*ek*p1+q(1)*q(5)*ek*p1+6*q(2)*q(4)*ek*p1+q(4
     . )*q(7)*ek+q(4)*q(5)*ek)/3.0+ej*ek*p1**2*(8*q(1)*q(2)*p1**4-(2
     . *q(1)*q(3)*ek*p1**3)-(8*q(1)*q(7)*p1**3)-(8*q(2)*q(6)*p1**3)-(30*
     . q(3)*q(6)*ek*p1**2)-(32*q(3)*q(4)*ek*p1**2)-(5*q(1)*q(2)*ek*p1**2
     . )+8*q(6)*q(7)*p1**2+14*q(1)*q(7)*ek*p1+15*q(2)*q(6)*ek*p1+7*q(1)*
     . q(5)*ek*p1+10*q(2)*q(4)*ek*p1+36*q(6)*q(7)*ek+50*q(4)*q(7)*ek+21*
     . q(5)*q(6)*ek+28*q(4)*q(5)*ek)/3.0
       t0=t0+2.0/3.0*el*ek**2*p1**2*(6*q(1)*q(2)*p1**4-(12*q(1)*q(7)*
     . p1**3)-(10*q(2)*q(6)*p1**3)-(6*q(1)*q(5)*p1**3)-(4*q(2)*q(4)*p1**
     . 3)-(4*q(3)*q(6)*ek*p1**2)-(4*q(3)*q(4)*ek*p1**2)+16*q(6)*q(7)*p1
     . **2+4*q(4)*q(7)*p1**2+6*q(5)*q(6)*p1**2+3*q(1)*q(7)*ek*p1+3*q(2)*
     . q(6)*ek*p1+3*q(1)*q(5)*ek*p1+3*q(2)*q(4)*ek*p1+7*q(6)*q(7)*ek+10*
     . q(4)*q(7)*ek+6*q(5)*q(6)*ek+9*q(4)*q(5)*ek)+4*te*(q(6)+q(4))*
     . (q(7)+q(5))*ek**4*p1**4-(1.0/3.0*re*ek**3*p1**2*(8*q(1)*q(7)*
     . p1**3+8*q(2)*q(6)*p1**3+8*q(1)*q(5)*p1**3+8*q(2)*q(4)*p1**3-(24*q
     . (6)*q(7)*p1**2)-(16*q(4)*q(7)*p1**2)-(16*q(5)*q(6)*p1**2)-(8*q(4)
     . *q(5)*p1**2)-(3*q(6)*q(7)*ek)-(3*q(4)*q(7)*ek)-(3*q(5)*q(6)*ek)-(
     . 3*q(4)*q(5)*ek)))
       bmn2(4,7)=t0-(1.0/3.0*de*(5*q(1)*q(3)*p1**3+3*q(3)*q(6)*p1**2
     . +8*q(3)*q(4)*p1**2-(4*q(1)*q(2)*p1**2)-(q(1)*q(7)*p1)-(3*q(2)*q(6
     . )*p1)-(7*q(2)*q(4)*p1)-(q(4)*q(7))))
c
c
```

```
      t0=2.0/3.0*el*ek**2*(2*q(1)**2*p1**6-(4*q(1)*q(6)*p1**5)+2*q(6
     . )**2*p1**4+8*q(2)**2*p1**4+3*q(1)*q(6)*ek*p1**3+3*q(1)*q(4)*ek*p1
     . **3-(16*q(2)*q(7)*p1**3)-(12*q(3)*q(7)*ek*p1**2)+q(6)**2*ek*p1**2
     . +5*q(4)*q(6)*ek*p1**2-(12*q(3)*q(5)*ek*p1**2)+4*q(4)**2*ek*p1**2+
     . 8*q(7)**2*p1**2+9*q(2)*q(7)*ek*p1+9*q(2)*q(5)*ek*p1+3*q(7)**2*ek+
     . 3*q(5)*q(7)*ek)-(1.0/6.0*re*ek**3*(16*q(1)*q(6)*p1**5+16*q(1)
     . *q(4)*p1**5-(16*q(6)**2*p1**4)-(16*q(4)*q(6)*p1**4)+48*q(2)*q(7)*
     . p1**3+48*q(2)*q(5)*p1**3-(3*q(6)**2*ek*p1**2)-(6*q(4)*q(6)*ek*p1
     . **2)-(3*q(4)**2*ek*p1**2)-(48*q(7)**2*p1**2)-(48*q(5)*q(7)*p1**2)
     . -(9*q(7)**2*ek)-(18*q(5)*q(7)*ek)-(9*q(5)**2*ek)))
      bmn2(5,5)=t0+ej*ek**2*(16*q(3)**2*p1**4-(3*q(1)**2*p1**4)+6*q(
     . 1)*q(6)*p1**3-(4*q(2)*q(3)*p1**3)-(28*q(3)*q(7)*p1**2)+q(6)**2*p1
     . **2+8*q(4)*q(6)*p1**2+4*q(4)**2*p1**2-(8*q(2)**2*p1**2)+20*q(2)*q
     . (7)*p1+4*q(7)**2)/3.0+2*te*ek**4*p1**2*((q(6)+q(4))**2*p1**2+
     . 3*(q(7)+q(5))**2)+he*ek**2*(7*q(3)**2*p1**2+q(1)**2*p1**2-(7*
     . q(2)*(2*q(3)*p1-q(2)))+2*q(1)*q(4)*p1+q(4)**2)/6.0
c
c
      t0=he*ek*p1*(2*q(1)*q(3)*p1**4-(10*q(3)*q(6)*p1**3)-(8*q(3)*q(
     . 4)*p1**3)-(8*q(1)*q(2)*p1**3)-(3*q(1)*q(3)*ek*p1**2)+6*q(1)*q(7)*
     . p1**2+8*q(2)*q(6)*p1**2-(3*q(3)*q(4)*ek*p1)+3*q(1)*q(2)*ek*p1+2*q
     . (6)*q(7)*p1+8*q(4)*q(7)*p1+3*q(2)*q(4)*ek)/3.0+ej*ek*p1**2*(8
     . *q(1)*q(2)*p1**4-(2*q(1)*q(3)*ek*p1**3)-(8*q(1)*q(7)*p1**3)-(8*q(
     . 2)*q(6)*p1**3)-(20*q(3)*q(6)*ek*p1**2)-(22*q(3)*q(4)*ek*p1**2)-(5
     . *q(1)*q(2)*ek*p1**2)+8*q(6)*q(7)*p1**2+13*q(1)*q(7)*ek*p1+14*q(2)
     . *q(6)*ek*p1+6*q(1)*q(5)*ek*p1+9*q(2)*q(4)*ek*p1+8*q(6)*q(7)*ek+21
     . *q(4)*q(7)*ek+2*q(5)*q(6)*ek+8*q(4)*q(5)*ek)/3.0
      bmn2(5,6)=t0+2.0/3.0*el*ek**2*p1**2*(6*q(1)*q(2)*p1**4-(10*q(1
     . )*q(7)*p1**3)-(12*q(2)*q(6)*p1**3)-(4*q(1)*q(5)*p1**3)-(6*q(2)*q(
     . 4)*p1**3)-(4*q(3)*q(6)*ek*p1**2)-(4*q(3)*q(4)*ek*p1**2)+16*q(6)*q
     . (7)*p1**2+6*q(4)*q(7)*p1**2+4*q(5)*q(6)*p1**2+3*q(1)*q(7)*ek*p1+3
     . *q(2)*q(6)*ek*p1+3*q(1)*q(5)*ek*p1+3*q(2)*q(4)*ek*p1+3*q(6)*q(7)*
     . ek+6*q(4)*q(7)*ek+2*q(5)*q(6)*ek+5*q(4)*q(5)*ek)+4*te*(q(6)+q
     . (4))*(q(7)+q(5))*ek**4*p1**4-(1.0/3.0*re*ek**3*p1**2*(8*q(1)*
     . q(7)*p1**3+8*q(2)*q(6)*p1**3+8*q(1)*q(5)*p1**3+8*q(2)*q(4)*p1**3-
     . (24*q(6)*q(7)*p1**2)-(16*q(4)*q(7)*p1**2)-(16*q(5)*q(6)*p1**2)-(8
     . *q(4)*q(5)*p1**2)-(3*q(6)*q(7)*ek)-(3*q(4)*q(7)*ek)-(3*q(5)*q(6)*
     . ek)-(3*q(4)*q(5)*ek)))-(2.0/3.0*fe*ek*p1*(q(1)*p1+q(4))*(q(3)
     . *p1-q(2)))
c
c
      t0=ej*ek*(4*q(1)**2*p1**6-(8*q(1)*q(6)*p1**5)+16*q(3)**2*ek*p1
     . **4-(3*q(1)**2*ek*p1**4)+4*q(6)**2*p1**4+12*q(2)**2*p1**4+13*q(1)
     . *q(6)*ek*p1**3+7*q(1)*q(4)*ek*p1**3-(4*q(2)*q(3)*ek*p1**3)-(24*q(
     . 2)*q(7)*p1**3)-(56*q(3)*q(7)*ek*p1**2)+4*q(6)**2*ek*p1**2+21*q(4)
     . *q(6)*ek*p1**2-(28*q(3)*q(5)*ek*p1**2)+14*q(4)**2*ek*p1**2-(8*q(2
     . )**2*ek*p1**2)+12*q(7)**2*p1**2+40*q(2)*q(7)*ek*p1+20*q(2)*q(5)*
     . ek*p1+12*q(7)**2*ek+8*q(5)*q(7)*ek)/3.0
       t0=t0+el*ek**2*(4*q(1)**2*p1**6-(20*q(1)*q(6)*p1**5)-(12*q(1)*
     . q(4)*p1**5)+16*q(6)**2*p1**4+12*q(4)*q(6)*p1**4+16*q(2)**2*p1**4+
     . 6*q(1)*q(6)*ek*p1**3+6*q(1)*q(4)*ek*p1**3-(64*q(2)*q(7)*p1**3)-(
     . 32*q(2)*q(5)*p1**3)-(24*q(3)*q(7)*ek*p1**2)+3*q(6)**2*ek*p1**2+12
     . *q(4)*q(6)*ek*p1**2-(24*q(3)*q(5)*ek*p1**2)+9*q(4)**2*ek*p1**2+48
     . *q(7)**2*p1**2+32*q(5)*q(7)*p1**2+18*q(2)*q(7)*ek*p1+18*q(2)*q(5)
     . *ek*p1+9*q(7)**2*ek+12*q(5)*q(7)*ek+3*q(5)**2*ek)/3.0
       t0=t0-(1.0/6.0*re*ek**3*(16*q(1)*q(6)*p1**5+16*q(1)*q(4)*p1**5
     . -(24*q(6)**2*p1**4)-(32*q(4)*q(6)*p1**4)-(8*q(4)**2*p1**4)+48*q(2
     . )*q(7)*p1**3+48*q(2)*q(5)*p1**3-(3*q(6)**2*ek*p1**2)-(6*q(4)*q(6)
     . *ek*p1**2)-(3*q(4)**2*ek*p1**2)-(72*q(7)**2*p1**2)-(96*q(5)*q(7)*
     . p1**2)-(24*q(5)**2*p1**2)-(9*q(7)**2*ek)-(18*q(5)*q(7)*ek)-(9*q(5
     . )**2*ek)))+he*ek*(24*q(3)**2*p1**4-(6*q(1)**2*p1**4)+12*q(1)*
     . q(6)*p1**3+4*q(2)*q(3)*p1**3+7*q(3)**2*ek*p1**2+q(1)**2*ek*p1**2-
     . (52*q(3)*q(7)*p1**2)+2*q(6)**2*p1**2+16*q(4)*q(6)*p1**2+8*q(4)**2
     . *p1**2-(22*q(2)**2*p1**2)+2*q(1)*q(4)*ek*p1-(14*q(2)*q(3)*ek*p1)+
     . 40*q(2)*q(7)*p1+q(4)**2*ek+7*q(2)**2*ek+6*q(7)**2)/6.0+2*te*
     . ek**4*p1**2*((q(6)+q(4))**2*p1**2+3*(q(7)+q(5))**2)
       bmn2(5,7)=t0+fe*ek*(5*q(3)**2*p1**2+q(1)**2*p1**2-(5*q(2)*(2*q
     . (3)*p1-q(2)))+2*q(1)*q(4)*p1+q(4)**2)/6.0
c
c
       t0=he*p1**2*(36*q(1)**2*p1**6-(72*q(1)*q(6)*p1**5)+16*q(3)**2*
     . ek*p1**4-(44*q(1)**2*ek*p1**4)+36*q(6)**2*p1**4+12*q(2)**2*p1**4+
     . 120*q(1)*q(6)*ek*p1**3+32*q(1)*q(4)*ek*p1**3-(24*q(2)*q(7)*p1**3)
     . +q(3)**2*ek**2*p1**2+7*q(1)**2*ek**2*p1**2-(52*q(3)*q(7)*ek*p1**2
     . )+24*q(6)**2*ek*p1**2+168*q(4)*q(6)*ek*p1**2-(20*q(3)*q(5)*ek*p1
     . **2)+100*q(4)**2*ek*p1**2-(12*q(2)**2*ek*p1**2)+12*q(7)**2*p1**2+
     . 14*q(1)*q(4)*ek**2*p1-(2*q(2)*q(3)*ek**2*p1)+40*q(2)*q(7)*ek*p1+
     . 16*q(2)*q(5)*ek*p1+7*q(4)**2*ek**2+q(2)**2*ek**2+8*q(7)**2*ek+4*q
     . (5)*q(7)*ek)/6.0
       t0=t0+ej*ek*p1**2*(24*q(1)**2*p1**6-(72*q(1)*q(6)*p1**5)-(24*q
     . (1)*q(4)*p1**5)+4*q(3)**2*ek*p1**4-(8*q(1)**2*ek*p1**4)+48*q(6)**
     . 2*p1**4+24*q(4)*q(6)*p1**4+8*q(2)**2*p1**4+60*q(1)*q(6)*ek*p1**3+
     . 44*q(1)*q(4)*ek*p1**3-(24*q(2)*q(7)*p1**3)-(8*q(2)*q(5)*p1**3)-(
     . 28*q(3)*q(7)*ek*p1**2)+24*q(6)**2*ek*p1**2+108*q(4)*q(6)*ek*p1**2
     . -(20*q(3)*q(5)*ek*p1**2)+76*q(4)**2*ek*p1**2-(3*q(2)**2*ek*p1**2)
     . +16*q(7)**2*p1**2+8*q(5)*q(7)*p1**2+20*q(2)*q(7)*ek*p1+14*q(2)*q(
     . 5)*ek*p1+8*q(7)**2*ek+8*q(5)*q(7)*ek+q(5)**2*ek)/3.0
       t0=t0+2.0/3.0*el*ek**2*p1**2*(8*q(1)**2*p1**6-(48*q(1)*q(6)*p1
     . **5)-(32*q(1)*q(4)*p1**5)+48*q(6)**2*p1**4+48*q(4)*q(6)*p1**4+8*q
     . (4)**2*p1**4+2*q(2)**2*p1**4+9*q(1)*q(6)*ek*p1**3+9*q(1)*q(4)*ek*
     . p1**3-(16*q(2)*q(7)*p1**3)-(12*q(2)*q(5)*p1**3)-(4*q(3)*q(7)*ek*
     . p1**2)+6*q(6)**2*ek*p1**2+21*q(4)*q(6)*ek*p1**2-(4*q(3)*q(5)*ek*
     . p1**2)+15*q(4)**2*ek*p1**2+16*q(7)**2*p1**2+16*q(5)*q(7)*p1**2+2*
     . q(5)**2*p1**2+3*q(2)*q(7)*ek*p1+3*q(2)*q(5)*ek*p1+2*q(7)**2*ek+3*
     . q(5)*q(7)*ek+q(5)**2*ek)
      t0=t0-(1.0/6.0*re*ek**3*p1**2*(48*q(1)*q(6)*p1**5+48*q(1)*q(4)
     . *p1**5-(96*q(6)**2*p1**4)-(144*q(4)*q(6)*p1**4)-(48*q(4)**2*p1**4
     . )+16*q(2)*q(7)*p1**3+16*q(2)*q(5)*p1**3-(9*q(6)**2*ek*p1**2)-(18*
     . q(4)*q(6)*ek*p1**2)-(9*q(4)**2*ek*p1**2)-(32*q(7)**2*p1**2)-(48*q
     . (5)*q(7)*p1**2)-(16*q(5)**2*p1**2)-(3*q(7)**2*ek)-(6*q(5)*q(7)*ek
     . )-(3*q(5)**2*ek)))+fe*p1**2*(8*q(3)**2*p1**4-(19*q(1)**2*p1**
     . 4)+42*q(1)*q(6)*p1**3+4*q(1)*q(4)*p1**3+4*q(2)*q(3)*p1**3+2*q(3)
     . **2*ek*p1**2+10*q(1)**2*ek*p1**2-(20*q(3)*q(7)*p1**2)+9*q(6)**2*
     . p1**2+60*q(4)*q(6)*p1**2+32*q(4)**2*p1**2-(9*q(2)**2*p1**2)+20*q(
     . 1)*q(4)*ek*p1-(4*q(2)*q(3)*ek*p1)+14*q(2)*q(7)*p1+10*q(4)**2*ek+2
     . *q(2)**2*ek+3*q(7)**2)/6.0
      bmn2(6,6)=t0+2*te*ek**4*p1**4*(3*(q(6)+q(4))**2*p1**2+(q(7)+q(
     . 5))**2)+de*p1**2*(q(3)**2*p1**2+7*q(1)**2*p1**2-(q(2)*(2*q(3
     . )*p1-q(2)))+14*q(1)*q(4)*p1+7*q(4)**2)/6.0
c
c
      t0=he*p1*(12*q(1)*q(2)*p1**5+2*q(1)*q(3)*ek*p1**4-(12*q(1)*q(7
     . )*p1**4)-(12*q(2)*q(6)*p1**4)-(26*q(3)*q(6)*ek*p1**3)-(24*q(3)*q(
     . 4)*ek*p1**3)-(16*q(1)*q(2)*ek*p1**3)+12*q(6)*q(7)*p1**3-(3*q(1)*q
     . (3)*ek**2*p1**2)+20*q(1)*q(7)*ek*p1**2+20*q(2)*q(6)*ek*p1**2+6*q(
     . 1)*q(5)*ek*p1**2+4*q(2)*q(4)*ek*p1**2-(3*q(3)*q(4)*ek**2*p1)+3*q(
     . 1)*q(2)*ek**2*p1+8*q(6)*q(7)*ek*p1+28*q(4)*q(7)*ek*p1+2*q(5)*q(6)
     . *ek*p1+8*q(4)*q(5)*ek*p1+3*q(2)*q(4)*ek**2)/3.0-(1.0/3.0*fe*
     . p1*(2*q(1)*q(3)*p1**4+10*q(3)*q(6)*p1**3+12*q(3)*q(4)*p1**3+5*q(1
     . )*q(2)*p1**3+4*q(1)*q(3)*ek*p1**2-(7*q(1)*q(7)*p1**2)-(7*q(2)*q(6
     . )*p1**2)-(2*q(2)*q(4)*p1**2)+4*q(3)*q(4)*ek*p1-(4*q(1)*q(2)*ek*p1
     . )-(3*q(6)*q(7)*p1)-(10*q(4)*q(7)*p1)-(4*q(2)*q(4)*ek))
      t0=t0+ej*ek*p1**2*(16*q(1)*q(2)*p1**4-(2*q(1)*q(3)*ek*p1**3)-(
     . 24*q(1)*q(7)*p1**3)-(24*q(2)*q(6)*p1**3)-(8*q(1)*q(5)*p1**3)-(8*q
     . (2)*q(4)*p1**3)-(28*q(3)*q(6)*ek*p1**2)-(30*q(3)*q(4)*ek*p1**2)-(
     . 5*q(1)*q(2)*ek*p1**2)+32*q(6)*q(7)*p1**2+8*q(4)*q(7)*p1**2+8*q(5)
     . *q(6)*p1**2+20*q(1)*q(7)*ek*p1+20*q(2)*q(6)*ek*p1+13*q(1)*q(5)*ek
     . *p1+15*q(2)*q(4)*ek*p1+16*q(6)*q(7)*ek+36*q(4)*q(7)*ek+8*q(5)*q(6
     . )*ek+21*q(4)*q(5)*ek)/3.0
      bmn2(6,7)=t0+2.0/3.0*el*ek**2*p1**2*(6*q(1)*q(2)*p1**4-(16*q(1
     . )*q(7)*p1**3)-(16*q(2)*q(6)*p1**3)-(10*q(1)*q(5)*p1**3)-(10*q(2)*
     . q(4)*p1**3)-(4*q(3)*q(6)*ek*p1**2)-(4*q(3)*q(4)*ek*p1**2)+32*q(6)
     . *q(7)*p1**2+16*q(4)*q(7)*p1**2+16*q(5)*q(6)*p1**2+6*q(4)*q(5)*p1
     . **2+3*q(1)*q(7)*ek*p1+3*q(2)*q(6)*ek*p1+3*q(1)*q(5)*ek*p1+3*q(2)*
     . q(4)*ek*p1+4*q(6)*q(7)*ek+7*q(4)*q(7)*ek+3*q(5)*q(6)*ek+6*q(4)*q(
     . 5)*ek)+4*te*(q(6)+q(4))*(q(7)+q(5))*ek**4*p1**4-(1.0/3.0*re
     . *ek**3*p1**2*(8*q(1)*q(7)*p1**3+8*q(2)*q(6)*p1**3+8*q(1)*q(5)*
     . p1**3+8*q(2)*q(4)*p1**3-(32*q(6)*q(7)*p1**2)-(24*q(4)*q(7)*p1**2)
     . -(24*q(5)*q(6)*p1**2)-(16*q(4)*q(5)*p1**2)-(3*q(6)*q(7)*ek)-(3*q(
     . 4)*q(7)*ek)-(3*q(5)*q(6)*ek)-(3*q(4)*q(5)*ek)))-(de*p1*(q(1)
     . *p1+q(4))*(q(3)*p1-q(2)))
c
c
      t0=he*(12*q(1)**2*p1**6-(24*q(1)*q(6)*p1**5)+48*q(3)**2*ek*p1
     . **4-(12*q(1)**2*ek*p1**4)+12*q(6)**2*p1**4+36*q(2)**2*p1**4+40*q(
     . 1)*q(6)*ek*p1**3+16*q(1)*q(4)*ek*p1**3+8*q(2)*q(3)*ek*p1**3-(72*q
     . (2)*q(7)*p1**3)+7*q(3)**2*ek**2*p1**2+q(1)**2*ek**2*p1**2-(156*q(
     . 3)*q(7)*ek*p1**2)+8*q(6)**2*ek*p1**2+56*q(4)*q(6)*ek*p1**2-(52*q(
     . 3)*q(5)*ek*p1**2)+36*q(4)**2*ek*p1**2-(44*q(2)**2*ek*p1**2)+36*q(
     . 7)**2*p1**2+2*q(1)*q(4)*ek**2*p1-(14*q(2)*q(3)*ek**2*p1)+120*q(2)
     . *q(7)*ek*p1+40*q(2)*q(5)*ek*p1+q(4)**2*ek**2+7*q(2)**2*ek**2+24*q
     . (7)**2*ek+12*q(5)*q(7)*ek)/6.0
      t0=t0+ej*ek*(8*q(1)**2*p1**6-(24*q(1)*q(6)*p1**5)-(8*q(1)*q(4)
     . *p1**5)+16*q(3)**2*ek*p1**4-(3*q(1)**2*ek*p1**4)+16*q(6)**2*p1**4
     . +8*q(4)*q(6)*p1**4+24*q(2)**2*p1**4+20*q(1)*q(6)*ek*p1**3+14*q(1)
     . *q(4)*ek*p1**3-(4*q(2)*q(3)*ek*p1**3)-(72*q(2)*q(7)*p1**3)-(24*q(
     . 2)*q(5)*p1**3)-(84*q(3)*q(7)*ek*p1**2)+8*q(6)**2*ek*p1**2+36*q(4)
     . *q(6)*ek*p1**2-(56*q(3)*q(5)*ek*p1**2)+25*q(4)**2*ek*p1**2-(8*q(2
     . )**2*ek*p1**2)+48*q(7)**2*p1**2+24*q(5)*q(7)*p1**2+60*q(2)*q(7)*
     . ek*p1+40*q(2)*q(5)*ek*p1+24*q(7)**2*ek+24*q(5)*q(7)*ek+4*q(5)**2*
     . ek)/3.0
      t0=t0+2.0/3.0*el*ek**2*(2*q(1)**2*p1**6-(16*q(1)*q(6)*p1**5)-(
     . 12*q(1)*q(4)*p1**5)+16*q(6)**2*p1**4+16*q(4)*q(6)*p1**4+2*q(4)**2
     . *p1**4+8*q(2)**2*p1**4+3*q(1)*q(6)*ek*p1**3+3*q(1)*q(4)*ek*p1**3-
     . (48*q(2)*q(7)*p1**3)-(32*q(2)*q(5)*p1**3)-(12*q(3)*q(7)*ek*p1**2)
     . +2*q(6)**2*ek*p1**2+7*q(4)*q(6)*ek*p1**2-(12*q(3)*q(5)*ek*p1**2)+
     . 5*q(4)**2*ek*p1**2+48*q(7)**2*p1**2+48*q(5)*q(7)*p1**2+8*q(5)**2*
     . p1**2+9*q(2)*q(7)*ek*p1+9*q(2)*q(5)*ek*p1+6*q(7)**2*ek+9*q(5)*q(7
     . )*ek+3*q(5)**2*ek)
      t0=t0-(1.0/6.0*re*ek**3*(16*q(1)*q(6)*p1**5+16*q(1)*q(4)*p1**5
     . -(32*q(6)**2*p1**4)-(48*q(4)*q(6)*p1**4)-(16*q(4)**2*p1**4)+48*q(
     . 2)*q(7)*p1**3+48*q(2)*q(5)*p1**3-(3*q(6)**2*ek*p1**2)-(6*q(4)*q(6
     . )*ek*p1**2)-(3*q(4)**2*ek*p1**2)-(96*q(7)**2*p1**2)-(144*q(5)*q(7
     . )*p1**2)-(48*q(5)**2*p1**2)-(9*q(7)**2*ek)-(18*q(5)*q(7)*ek)-(9*q
     . (5)**2*ek)))+fe*(32*q(3)**2*p1**4-(9*q(1)**2*p1**4)+14*q(1)*q
     . (6)*p1**3-(4*q(1)*q(4)*p1**3)-(4*q(2)*q(3)*p1**3)+10*q(3)**2*ek*
     . p1**2+2*q(1)**2*ek*p1**2-(60*q(3)*q(7)*p1**2)+3*q(6)**2*p1**2+20*
     . q(4)*q(6)*p1**2+8*q(4)**2*p1**2-(19*q(2)**2*p1**2)+4*q(1)*q(4)*ek
     . *p1-(20*q(2)*q(3)*ek*p1)+42*q(2)*q(7)*p1+2*q(4)**2*ek+10*q(2)**2*
     . ek+9*q(7)**2)/6.0
```

```
      bmn2(7,7)=t0+2*te*ek**4*p1**2*((q(6)+q(4))**2*p1**2+3*(q(7)+q(
     . 5))**2)+de*(7*q(3)**2*p1**2+q(1)**2*p1**2-(7*q(2)*(2*q(3)*p1
     . -q(2)))+2*q(1)*q(4)*p1+q(4)**2)/6.0
c
do 100 ii=1,7
do 100 jj=ii,7
 100 bmn2(jj,ii)=bmn2(ii,jj)
return
end
c
c
c
subroutine sbeamk(bmk,ek1)
c
implicit double precision (a-h,o-z)
c
common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
dimension bmk(7,7)
do 10 ii=1,7
do 10 jj=1,7
  10 bmk(jj,ii)=0.0d0
c
      bmk(2,2)=ae
c
      bmk(5,5)=he*ek1**2
c
      bmk(5,7)=he*ek1**2+fe*ek1
c
      bmk(7,7)=he*ek1**2+fe*2*ek1+de
c
      bmk(4,4)=9*fs*ek1**2+6*ds*ek1+as
c
      bmk(4,6)=9*fs*ek1**2+6*ds*ek1+as
c
      bmk(6,6)=9*fs*ek1**2+6*ds*ek1+as
c
do 100 ii=1,7
do 100 jj=ii,7
 100 bmk(jj,ii)=bmk(ii,jj)
return
end
c
c
c
        subroutine sbmn1(q,bmn1,ek)
```

```
c
c Note that k1 appears as 'ek' in this subroutine
c
c The equations in this subroutine were generated by MACSYMA.
c
implicit double precision (a-h,o-z)
c
common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
dimension bmn1(7,7),q(7)
c
      bmn1(2,2)=3*q(2)*ae
c
      bmn1(2,4)=ae*q(4)
c
      bmn1(2,5)=3*he*(q(7)+q(5))*ek**2+3*fe*q(7)*ek
c
      bmn1(2,6)=0.0
c
      bmn1(2,7)=3*he*(q(7)+q(5))*ek**2+3*fe*(2*q(7)+q(5))*ek+3*de*q(7)
c
      bmn1(4,4)=q(2)*ae
c
      bmn1(5,5)=3*q(2)*he*ek**2
c
      bmn1(5,7)=3*q(2)*he*ek**2+3*q(2)*fe*ek
c
      bmn1(7,7)=3*q(2)*he*ek**2+6*q(2)*fe*ek+3*q(2)*de
c
do 100 ii=1,7
do 100 jj=ii,7
 100 bmn1(jj,ii)=bmn1(ii,jj)
return
end
c
c
c
subroutine sbmn2(q,bmn2,ek)
c
c note that k1 appears as 'ek' in this subroutine
c
c the equations in this subroutine were generated by macsyma.
c
implicit double precision (a-h,o-z)
c
common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
```

```
dimension bmn2(7,7),q(7)
c
c
      bmn2(2,2)=7.0/6.0*he*(q(7)+q(5))**2*ek**2+5.0/3.0*fe*q(7)*
     . (q(7)+q(5))*ek+7.0/6.0*de*q(7)**2+ae*(q(4)**2+3*q(2)**2)
     . /2.0
c
bmn2(2,4)=q(2)*ae*q(4)
c
      bmn2(2,5)=7.0/3.0*q(2)*he*(q(7)+q(5))*ek**2+5.0/3.0*q(2)*fe
     . *q(7)*ek

c
      bmn2(2,7)=7.0/3.0*q(2)*he*(q(7)+q(5))*ek**2+5.0/3.0*q(2)*fe
     . *(2*q(7)+q(5))*ek+7.0/3.0*q(2)*de*q(7)
c
      bmn2(4,4)=he*(q(7)+q(5))**2*ek**2/6.0+fe*q(7)*(q(7)+q(5))*
     . ek/3.0+de*q(7)**2/6.0+ae*(3*q(4)**2+q(2)**2)/2.0
c
      bmn2(4,5)=he*q(4)*(q(7)+q(5))*ek**2/3.0+fe*q(4)*q(7)*ek/
     . 3.0
c
      bmn2(4,7)=he*q(4)*(q(7)+q(5))*ek**2/3.0+fe*q(4)*(2*q(7)+q(
     . 5))*ek/3.0+de*q(4)*q(7)/3.0
c
      bmn2(5,5)=3.0/2.0*re*(q(7)+q(5))**2*ek**4+2*el*q(7)*(q(7)+
     . q(5))*ek**3+4.0/3.0*ej*q(7)**2*ek**2+he*(q(4)**2+7*q(2)**
     . 2)*ek**2/6.0
c
      bmn2(5,7)=3.0/2.0*re*(q(7)+q(5))**2*ek**4+el*(q(7)+q(5))*(
     . 3*q(7)+q(5))*ek**3+4.0/3.0*ej*q(7)*(3*q(7)+2*q(5))*ek**2+he
     . *ek*((q(4)**2+7*q(2)**2)*ek+6*q(7)**2)/6.0+fe*(q(4)**2+5*q(
     . 2)**2)*ek/6.0
c
      bmn2(7,7)=3.0/2.0*re*(q(7)+q(5))**2*ek**4+2*el*(q(7)+q(5))
     . *(2*q(7)+q(5))*ek**3+4.0/3.0*ej*(6*q(7)**2+6*q(5)*q(7)+q(5)**
     . 2)*ek**2+he*ek*((q(4)**2+7*q(2)**2)*ek+12*q(7)*(2*q(7)+q(5)))
     . /6.0+fe*(2*(q(4)**2+5*q(2)**2)*ek+9*q(7)**2)/6.0+de*(q(4
     . )**2+7*q(2)**2)/6.0
c
do 100 ii=1,7
do 100 jj=ii,7
 100 bmn2(jj,ii)=bmn2(ii,jj)
return
end
```

Bibliography

1. Belytschko, Ted and Lawrence W. Glaum. "Applications oof Higher Order Corotational Stretch Theories to Nonlinear Finite Element Analysis," *Computers and Structures*, *10*:175–182 (1979).
2. Brockman, Robert A. *Magna: A Finite Element Program for the Materially and Geometrically Nonlinear Analysis of Three-Dimensional Structures Subjected to Static and Transient Loading*. Technical Report, January 1981.
3. Cook, Robert D., et al. *Concepts and Applications of Finite Elements Analysis*. New York: John Wiley and Sons, 1989.
4. Crisfield, M.A. "A Fast Incremental/Iterative Solution Procedure that Handles Snap Through," *Computers and Structures*, *13*:55–62 (1981).
5. DaDeppo, D.A. and R. Schmidt. "Nonlinear Theory of Arches with Transverse Shear Deformation and Rotary Inertia," *Industrial Mathematics*, *21*:33–49 (1971).
6. DaDeppo, D.A. and R. Schmidt. "Large Deflections and Stability of Hingeless Circular Arches Under Interacting Loads," *Journal of Applied Mechanics*, 989–994 (December 1974).
7. DaDeppo, D.A. and R. Schmidt. "Instability of Clamped–Hinged Circular Arches Subjected to a Point Load," *Transactions of the ASME*, *42*:894–896 (December 1975).
8. Dennis, Scott T. *Large Displacement and Rotational Formulation for Laminated Cylindrical Shells Including Parabolic Transverse Shear*. PhD dissertation, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, May 1988.
9. Epstein, Marcello and David W. Murray. "Large Deformation In–Plane Analysis of Elastic Beams," *Computers and Structures*, *6*:1–9 (1976).
10. Huddleston, J. V. "Finite Deflections and Snap–Through of High Circular Arches," *Journal of Applied Mechanics*, 763–769 (December 1968).
11. Mindlin, R.D. "Influence of Rotary Inertia and Shear on Flexural Motions of Isotropic, Elastic Plates," *Journal of Applied Mechanics*, *18*:31–38 (March 1951).
12. Minguet, Pierre and John Dugundji. "Experiments and Analysis for Composite Blades Under Large Deflections Part I: Static Behavior," *AIAA Journal*, *28*:1573–1579 (September 1990).
13. Palazotto, Anthony N. and Scott T. Dennis. *Nonlinear Analysis of Shell Structures*. Washington, D.C.: American Institute of Aeronautics and Astronautics, 1992.
14. Ramm, E. "Strategies for Tracing the Nonlinear Response Near Limit Points." *Nonlinear Finite Element Analysis in Structural Mechanics* edited by W Wunderlich, et al., New York: Springer–Verlag, 1981.
15. Reddy, J. N. "Two-Dimensional Theories of Plates." *Finite Element Analysis for Engineering Design* edited by J.N. et al Reddy, Springer–Verlag, 1988.

16. Reddy, J.N. *Energy and Variational Methods in Applied Mechanics*. New York: John Wiley and Sons, 1984.

17. Reddy, J.N. "A Simple Higher-Order Theory for Laminated Composite Plates," *Journal of Applied Mechanics*, *51*:745-752 (December 1984).

18. Reissner, Eric. "The Effect of Transverse Shear Deformation on the Bending of Elastic Plates," *Journal of Applied Mechanics*, A-69-A-77 (June 1965).

19. Riks, E. "An Incremental Approach to the Solution of Snapping and Buckling Problems," *International Journal of Solids and Structures*, *15*:529-551 (1979).

20. Saada, Adel S. *Elasticity Theory and Applications*. Malabar, FL: Krieger, 1989.

21. Sabir, A.B. and A.C. Lock. "Large Deflexion, Geometrically Non-Linear Finite Element Analysis of Circular Arches," *International Journal of Mechanical Sciences*, *15*:37-47 (1973).

22. Smith, R. A. *Higher-Order Thickness Expansions for Cylindrical Shells*. PhD dissertation, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 1991.

23. Tsai, C. T and A. N. Palazotto. "Nonlinear and Multiple Snapping Responses of Cylindrical Panels Comparing Displacement Control and Riks Method," *Computers and Structures*, *41*(4):605-610 (1991).

Captain Stephen George Creaghan was born on September 27, 1957, in Baltimore, Maryland. He graduated from Pikesville Senior High School, in Pikesville, Maryland, in 1975. He enlisted in the U.S. Air Force in 1978. In 1988, he received a B.S. Civil Engineering degree, with high honors, from the University of Florida. He was commissioned a Second Lieutenant on September 29, 1988. Captain Creaghan was assigned to the 354th Civil Engineering Squadron, Myrtle Beach Air Force Base and accompanied the 354th Tactical Fighter Wing to Saudi Arabia for Desert Shield/Storm. He was assigned to the Air Force Institute of Technology in May 1991.